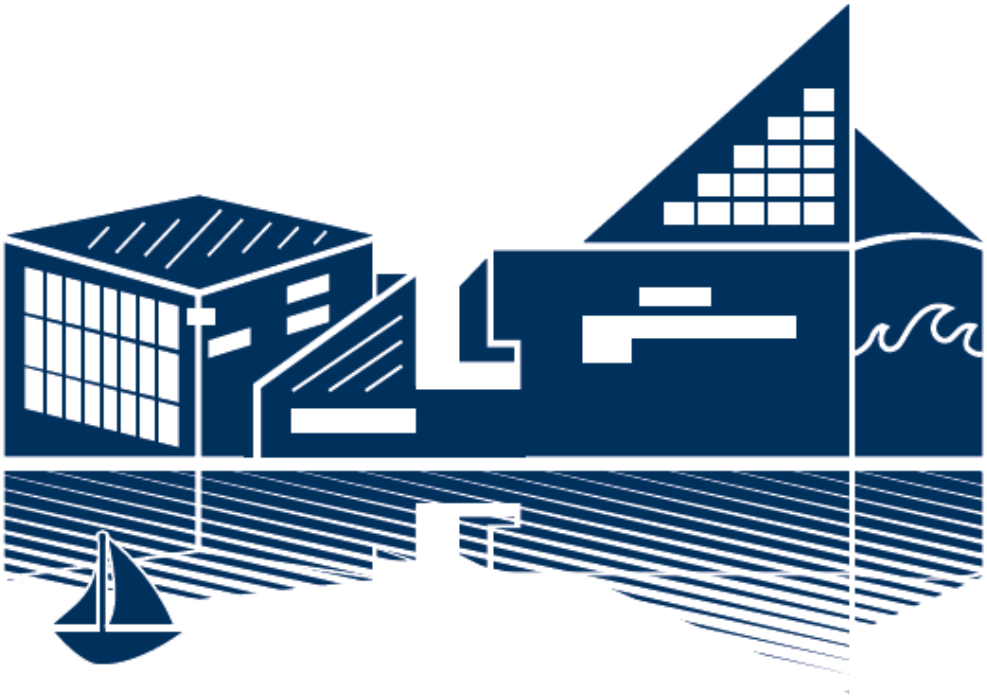


The 52nd Annual Meeting of the
Association for Computational Linguistics

Proceedings of the Conference
Volume 1: Long Papers

ACL 2014
June 22–27
Baltimore

Platinum Level Sponsor:

Gold Level Sponsors:

Silver Level Sponsors:

Bronze Level Sponsors:

Supporters:

Order copies of this and other ACL proceedings from:

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

ISBN 978-1-937284-72-5

Preface: General Chair

I remember with great fondness the first ACL Conference I attended 20 years ago in Las Cruces, New Mexico. Some things have changed: papers presented there that I considered interesting or inconsequential have switched positions in my personal ranking as I learned more and more about our field; single sessions have long been replaced by parallel sessions to accommodate an ever increasing number of research contributions; the number of associated workshops and posters has mushroomed beyond anyone's dream. Almost without noticing, we transitioned from small conferences of a few hundred to conferences that bring together 1000 plus participants from all over the world. Our field has matured significantly attracting the attention of not only a handful of academics, but successful industries and Research Labs as well. Some things have stayed the same though: ACL continues to be the pre-eminent conference in our field and the best place to meet and make like-minded friends, discuss tantalizing tricks that you can learn about only in face-to-face communication settings, and celebrate the results we get.

ACL Conferences are never possible without the dedication and hard work of many people. Because ACL'2014 is no exception to this, I would like to thank each and every person who has volunteered their time to make the event possible.

Priscilla Rasmussen, the ACL Business Manager, and the ACL Executive Committee (Haifeng Wang, Gertjan van Noord, Graeme Hirst, Dragomir Radev, Renata Vieira, Jian Su, Min-Yen Kan, Stephen Clark, and Hal Daumé III) have been instrumental in setting ACL'2014 in motion and in guiding the ACL'2014 committee along the path from concept to execution. Without the collective memory and hands-on guidance of the committee, an ACL conference will never happen.

The ACL'2014 Committee did a fantastic job making this conference possible. The committee covered a lot of ground from logistics to paper selection, to co-located event selection and publishing. The masterminds of all these intertwining tasks were: Kristina Toutanova and Hua Wu (Program Committee Chairs); David Yarowsky (Local Arrangements Chair); Jill Burstein and Lluís Màrquez (Workshop Chairs); Alex Fraser and Yang Liu (Tutorial Chairs), Alexander Koller and Miyao Yusuke (Publication Chairs); Ekaterina Kochmar, Annie Louis, and Svitlana Volkova (Student Research Workshop Chairs); Bill Byrne and Jordan Boyd-Graber (Faculty Advisors for the Student Workshop Chairs); Kalina Bontcheva and Zhu Jingbo (Demonstration Chairs); and Jason Riesa (Publicity Chair). The Program Chairs were also instrumental in selecting our outstanding invited speakers: Corinna Cortez (Google) and Zoran Popovic (University of Washington).

I am also grateful to our sponsors for their generous contributions, without which the conference would become prohibitively expensive for the next generation of computational linguistic researchers: Baidu (Platinum Sponsor); Bloomberg, Google, Microsoft, Nuance, and Yahoo Labs (Gold Sponsors); Information Sciences Institute and Xerox Research Center Europe (Silver Sponsors); Brandeis University, Facebook, and Yandex (Bronze Sponsors); and IBM Research and the University of Washington (Supporters).

Finally, I would like to express my appreciation to the area chairs, workshop organizers, tutorial presenters, and reviewers. And to all the ACL'2014 attendees. This is *your* conference; make the most of it!

Welcome to ACL'2014!

The ACL'2014 General Chair
Daniel Marcu, Information Sciences Institute, USC

Preface: Program Committee Co-Chairs

Welcome to the 2014 Conference of the Association for Computational Linguistics! This year ACL received 572 long paper submissions and 551 short paper submissions. Of the long papers, 146 were accepted for presentation at ACL — 95 as oral, and 51 as poster presentations. 139 short papers were accepted — 51 as oral, and 88 as poster presentations.

The submissions were reviewed under different categories and using different review forms for empirical/data-driven, theoretical, applications/tools, resources/evaluation, and survey papers. For the short papers we additionally used a negative results category and were glad to see that the community is becoming more open to enabling the publication of useful negative results.

Based on feedback from prior years, this year we organized the posters in two large poster sessions to accommodate the growing number of high-quality submissions accepted in poster presentation format. We hope attendees and authors will benefit from this additional time to present and discuss ideas. Another innovation we are experimenting with this year is to optimize the conference schedule based on feedback from attendees on the talks they would like to see. We collected attendee responses using a scheduling survey developed with the help of David Yarowsky and Svitlana Volkova (thanks to the 338 volunteers who completed the survey!), and we optimized the conference schedule to assign popular sessions to large conference rooms, and to reduce the chance that two talks that an attendee is interested in are scheduled at the same time. Additionally, as in NAACL 2013, all talks will be recorded and made available for future viewing.

ACL 2014 will have two distinguished invited speakers. Corinna Cortes (Head of Google Research, NY) and Zoran Popović (Director of Center for Game Science, University of Washington).

There are many individuals to thank for their contributions to ACL 2014. We would like to thank the thirty three area chairs for their hard work on recruiting reviewers, leading the discussion process, and carefully ranking the submissions. We would like to thank Mark Dredze for developing and sharing a reviewer assignment tool, that was used at ACL this year. It was applied to ACL reviewing with the help of Jiang Guo and the area chairs who provided feedback at several stages of the process. We would also like to thank the seven hundred and seventy nine reviewers and seventy two secondary reviewers on whose efforts we depend to select high-quality and timely scientific work. This year we specifically acknowledged around 14% of the reviewers who went the extra mile and provided extremely helpful to the area chairs and authors reviews (their names are marked with a * in the organization section of the proceedings). The ACL coordinating committee members, including Dragomir Radev, Jian Su, Graeme Hirst, Hal Daumé III, Chris Callison-Burch, and Haifeng Wang were very helpful on various issues relating to the organization. We would like to thank the prior conference chairs Jason Eisner, Hal Daumé, Lucy Vanderwende, Jian Su, Rada Mihalcea, Marius Pasca, Pascale Fung, and Massimo Poesio for their advice. We are very grateful for the guidance and support of the general chair Daniel Marcu, to the ACL Business Manager Priscilla Rasmussen who knew practically everything, to the local chair David Yarowsky, the publication chairs Yusuke Miyao and Alexander Koller, and to Matt Post who stepped in to handle the conference handbook. We would also like to thank Jiang Guo who helped with reviewer assignment and numerous other tasks. Rich Gerber from Softconf was extremely responsive to all of our requests, and we are grateful for that.

We hope you will enjoy ACL 2014 in Baltimore!

ACL 2014 Program Co-Chairs
Kristina Toutanova, Microsoft Research
Hua Wu, Baidu

Organizing Committee

General Conference Chair

Daniel Marcu, ISI/USC

Program Committee Chairs

Kristina Toutanova, Microsoft Research
Hua Wu, Baidu

Local Arrangements Chair

David Yarowsky, Johns Hopkins University

Workshop Chairs

Jill Burstein, Educational Testing Service
Lluís Màrquez, Universitat Politècnica de Catalunya

Tutorial Chairs

Alex Fraser, University of Munich
Yang Liu, Tsinghua University

Publications Chairs

Alexander Koller, University of Potsdam
Yusuke Miyao, National Institute of Informatics Japan
We gratefully acknowledge the support of Ulla Behr, Anna Lukowiak, and Deyan Ginev.

Student Research Workshop Chairs

Ekaterina Kochmar, University of Cambridge
Annie Louis, University of Pennsylvania
Svitlana Volkova, Johns Hopkins University

Faculty Advisors for the Student Research Workshop

Bill Byrne, University of Cambridge
Jordan Boyd-Graber, University of Maryland

Demonstrations Chairs

Kalina Bontcheva, University of Sheffield
Jingbo Zhu, Northeastern University China

Publicity Chair

Jason Riesa, Google

Business Manager

Priscilla Rasmussen

Reviewing Coordinators

Mark Dredze, Johns Hopkins University
Jiang Guo, Harbin Institute of Technology

Conference Handbook Editor

Matt Post, Johns Hopkins University

Local Guide Editors

Juri Ganitkevitch, Johns Hopkins University
Ann Irvine, Johns Hopkins University

Student Volunteer Chair

Yuan Cao, Johns Hopkins University

Entertainment Chairs

Matt Gormley, Johns Hopkins University
Naomi Saphra, Johns Hopkins University

Graphic Design

Jeff Hayes, Jeff-Hayes.com
Meg Mitchell, Microsoft Research
Keisuke Sakaguchi, Johns Hopkins University

Local JHU/CLSP Administration

Laura Graham
Ruth Scally

Program Committee

Program Committee Chairs

Kristina Toutanova, Microsoft Research
Hua Wu, Baidu

Area Chairs

Klinton Bicknell, Northwestern University
Asli Celikyilmaz, Microsoft Research
Micha Elsner, Ohio State University
Michael Strube, HITS gGmbH
Jacob Eisenstein, Georgia Institute of Technology
Jun Zhao, Chinese Academy of Sciences
Barbara Di Eugenio, University of Illinois at Chicago
Ting Liu, Harbin Institute of Technology
Marius Pasca, Google Research
Mihai Surdeanu, University of Arizona
Nicoletta Calzolari, Institute of Computational Linguistics of the National Research Council
Nianwen Xue, Brandeis University
Eneko Agirre, University of the Basque Country
Kevin Duh, Nara Institute of Science and Technology
Jason Eisner, Johns Hopkins University
Colin Cherry, National Research Council, Canada
Niyu Ge, IBM Research
Liang Huang, City University of New York
Yanjun Ma, Baidu
Michel Galley, Microsoft Research
Brian Roark, Google Research
Mamoru Komachi, Tokyo Metropolitan University
Miles Osborne, Johns Hopkins University
Alan Ritter, Carnegie Mellon University
Iryna Gurevych, Technische Universität Darmstadt
Percy Liang, Stanford University
Shiqi Zhao, Baidu
Yang Liu, University of Texas at Dallas
Scott Wen-tau Yih, Microsoft Research
Ciprian Chelba, Google Research
Xavier Carreras, Universitat Politecnica de Catalunya
Slav Petrov, Google Research
Grzegorz Kondrak, University of Alberta

Primary Reviewers

Reviewers who are acknowledged by the program committee for providing one or more outstanding reviews are marked with "*".

Ahmed Abbasi, Omri Abend, Gilles Adda*, Željko Agić, Jan Alexandersson, Enrique Alfonseca, Afra Alishahi, Alexander Allauzen, Yasemin Altun, Ion Androutsopoulos*, Gabor Angeli*, Masahiro Araki, Yuki Arase, Ron Artstein, Yoav Artzi*, Jordi Atserias Batalla, Giuseppe Attardi,

Michael Auli

Olga Babko-Malaya, Anton Bakalov*, Timothy Baldwin*, Kalika Bali, Miguel Ballesteros, David Bamman, Rafael E. Banchs, Carmen Banea, Srinivas Bangalore, Mohit Bansal, Marco Baroni, Loïc Barrault, Anabela Barreiro, Regina Barzilay, Roberto Basili, John Bateman, Frederic Bechet, Steve Beet, Núria Bel, Kedar Bellare, Anja Belz*, Jose Miguel Benedi, Jonathan Berant*, Taylor Berg-Kirkpatrick, Sabine Bergler, Shane Bergsma*, Nicole Beringer, Laurent Besacier, Steven Bethard*, Chandra Bhagavatula, Suma Bhat, Pushpak Bhattacharyya, Chris Biemann*, Ann Bies, Graeme Blackwood, Phil Blunsom, Gemma Boleda, Danushka Bollegala, Francis Bond, Kalina Bontcheva, Stefano Borgo, Antal van den Bosch, Alexandre Bouchard, Jordan Boyd-Graber, Johan Boye, S.R.K. Branavan, António Branco*, Chris Brew, Ted Briscoe, Chris Brockett, Julian Brooke, Paul Buitelaar, Razvan Bunescu, Wray Buntine, David Burkett, Stephan Busemann, Bill Byrne

Elena Cabrio, Chris Callison-Burch, Marie Candito, Cornelia Caragea, Sandra Carberry, Jesus Cardeńosa, Giuseppe Carenini, Marine Carpuat, Xavier Carreras, John Carroll, Francisco Casacuberta, Tommaso Caselli, Taylor Cassidy, Vittorio Castelli, Daniel Cer, Christophe Cerisara, Soumen Chakrabarti, Nathanael Chambers, Yee Seng Chan, Kai-Wei Chang, Wanxiang Che, Ciprian Chelba, Boxing Chen, Hsin-Hsi Chen, John Chen, Chen Chen, Zheng Chen, Xueqi Cheng, Jackie Chi Kit Cheung, David Chiang, Laura Chiticariu*, Yejin Choi*, Monojit Choudhury, Grzegorz Chrupała, Jennifer Chu-Carroll*, Cindy Chung, Philipp Cimiano, Stephen Clark*, Ann Clifton, Shay B. Cohen, Trevor Cohn, Nigel Collier, Gao Cong, John Conroy, Paul Cook, Bonaventura Coppola, Anna Corazza, Mark Core, Marta R. Costa-jussà, Danilo Croce, Paul Crook, Xiaodong Cui, Iria da Cunha*

Beatrice Daille, Robert Daland, Hoa Trang Dang, Dipanjan Das*, Pradipto Das, Munmun De Choudhury, John DeNero, Rodolfo Delmonte, Jean-Yves Delort, Estelle Delpech, Vera Demberg, Pascal Denis, Michael Denkowski, Leon Derczynski, Jacob Devlin, Giuseppe Di Fabbrizio, Mona Diab, Markus Dickinson, Brian Dillon, Marco Dinarelli, Stefanie Dipper*, Bill Dolan*, Doug Downey, Mark Dras, Mark Dredze, Markus Dreyer, Gregory Druck, Lan Du, Jinhua Du, Ewan Dunbar*, Greg Durrett*, Chris Dyer, Marc Dymetman, Myroslava Dzikovska

Kurt Eberle, Judith Eckle-Kohler*, Koji Eguchi, Yo Ehara, Patrick Ehlen, Vladimir Eidelman, Andreas Eisele, Jason Eisner, Michael Elhadad, Ahmad Emami, Klaus-Peter Engelbrecht, Katrin Erk*, Andrea Esuli

Angela Fahrni, Mauro Falcone, James Fan, Hui Fang, Faisal Farooq, Manaal Faruqui, Benoit Favre, Afsaneh Fazly*, Anna Feldman, Christiane Fellbaum, Minwei Feng, Raquel Fernandez, Katja Filippova, Andrew Finch, Darja Fišer, Margaret Fleck, Mikel Forcada, Karën Fort, George Foster, Jennifer Foster, James Foulds*, Gil Francopoulo, Anette Frank*, Michael Frank, Stella Frank, Reva Freedman*, Markus Freitag

Evgeniy Gabrilovich*, Robert Gaizauskas*, Michael Gamon, Kuzman Ganchev, Sudeep Gandhe, Juri Ganitkevitch, Yue Gao, Wei Gao, Claire Gardent, Matt Gardner, Guillermo Garrido, Albert Gatt, Maria Gavrilidou, Josef van Genabith, Dmitriy Genzel, Kallirroi Georgila, Matthew Gerber, Daniel Gildea, Alastair Gill, Jennifer Gillenwater*, Kevin Gimpel*, Adrià de Gispert, Alfio Gliozzo, Yoav Goldberg*, Dan Goldwasser, Matthew R. Gormley*, Cyril Goutte*, Carlos Gómez-Rodríguez, Joao Graca, Brigitte Grau, Nancy Green, Spence Green*, Edward Grefenstette*, Gregory Grefenstette*, Justin Grimmer, Ralph Grishman, Marco Guerini, Camille Guinaudeau, Jiafeng Guo, Weiwei Guo, Yuhong Guo, Sonal Gupta, Iryna Gurevych, Joakim Gustafson

Ben Hachey*, Barry Haddow, Masato Hagiwara*, Dilek Hakkani-Tur, John Hale, David Hall*, Keith Hall, Bo Han*, Xianpei Han, Kazi Saidul Hasan, Saša Hasan, Mark Hasegawa-Johnson*, Hany Hassan, Wei He, Zhongjun He, Yifan He, Xiaodong He, Kenneth Heafield, Ulrich Heid, James Henderson*, John Henderson, Tsutomu Hirao, Keikichi Hirose, Graeme Hirst, Anna Hjal-

marsson, Hieu Hoang, Julia Hockenmaier, Johannes Hoffart, Kristy Hollingshead, Helmut Horacek, Ales Horak, Chiori Hori, Eduard Hovy, Estevam Hruschka, Shu-Kai Hsieh, Bo-June (Paul) Hsu, Yuening Hu, Fei Huang, Fei Huang, Ruihong Huang, Liang Huang, Xuanjing Huang, Zhongqiang Huang, Rebecca Hwa*

Nancy Ide, Ryu Iida, Shajith Ikbal, Iustina Ilisei, Diana Inkpen, Kentaro Inui, Ann Irvine, Hitoshi Isahara

Jagadeesh Jagarlamudi, Rahul Jha, Yangfeng Ji, Jing Jiang, Richard Johansson, Mark Johnson, Arne Jonsson

Kyo Kageura, Ozlem Kalinli, Min-Yen Kan*, Pallika Kanani, Evangelos Kanoulas, Kyoko Kanzaki, Damianos Karakos, Hisashi Kashima, Hideto Kazawa, Simon Keizer, Frank Keller, Emre Kiciman*, Bernd Kiefer, Jin-Dong Kim, Jungi Kim, Seungyeon Kim, Su Nam Kim, Tracy Holloway King, Brian Kingsbury, Chunyu Kit, Ioannis Klapaftis, Alexandre Klementiev, Kevin Knight, Alistair Knott, Philipp Koehn, Varada Kolhatkar, Oleksandr Kolomiyets, Kazunori Komatani, Grzegorz Kondrak, Terry Koo, Stefan Kopp, Valia Kordoni, Anna Korhonen, Zornitsa Kozareva, Jayant Krishnamurthy, Marco Kuhlmann*, Roland Kuhn, Shankar Kumar, A Kumaran, Polina Kuznetsova*, Sandra Kübler, Tom Kwiatkowski

Wai Lam, Mathias Lambert, Patrik Lambert, Phillippe Langlais*, Guy Lapalme, Romain Laroche, Dominique Laurent, Joseph Le Roux, Sungjin Lee, Oliver Lemon, Alessandro Lenci, Chee Wee Leong, Gina-Anne Levow, Roger Levy, Yan Li, Wenjie Li, Hang Li, Haizhou Li, Haibo Li, Lishuang Li, Qi Li, Shoushan Li, Mu Li, Peng Li, Rui Li, Maria Liakata, Chin-Yew Lin, Dekang Lin, Shou-de Lin, Xiao Ling, Ken Litkowski, Fei Liu, Jing Liu, Kang Liu, Lemao Liu, Bing Liu, Chang Liu, Yang Liu, Zhiyuan Liu, Qun Liu, Yiqun Liu, Eduardo Lleida Solano, Adam Lopez*, Oier Lopez de Lacalle, Ramon Lopez-Cozar, Annie Louis*, Wei Lu, Michael Lucas, Marco Lui, Franco M. Luque, Yajuan Lv

Klaus Macherey, Nitin Madnani*, Bente Maegaard, Mathew Magimai Doss, Andreas Maier, Inderjeet Mani, Joseph Mariani, Marie-Catherine de Marneffe, Erwin Marsi*, James H. Martin, Scott Martin*, David Martinez, André F. T. Martins*, Yuichiroh Matsubayashi, Yuji Matsumoto, Takuya Matsuzaki, Arne Mauser, Jon May, Lluís Màrquez, Diana McCarthy, David McClosky*, Kathy McCoy, Tara McIntosh*, Paul McNamee, Susan McRoy, Edgar Meij, Yelena Mejova, Gerard de Melo, Arul Menezes, Fandong Meng, Florian Metze, Christian M. Meyer*, Adam Meyers, Haitao Mi, Rada Mihalcea, Tomas Mikolov, Bonan Min, Yasuhiro Minami, Zhao-Yan Ming, Margaret Mitchell, Marie-Francine Moens, Saif Mohammad, Behrang Mohit, Karo Moilanen, Monica Monachini, Christian Monson, Manuel Montes, Christof Monz, Robert Moore*, Roser Morante, Alessandro Moschitti*, Dana Movshovitz-Attias*, Yugo Murawaki, Smaranda Muresan, Sunghyon Myaeng

Seiichi Nakagawa, Preslav Nakov*, Ramesh Nallapati, Jason Naradowsky*, Tahira Naseem, Vivi Nastase*, Borja Navarro, Roberto Navigli, Adeline Nazarenko, Mark-Jan Nederhof, Matteo Negri*, Ani Nenkova, John Nerbonne, Graham Neubig, Guenter Neumann, Hwee Tou Ng, Vincent Ng, Dong Nguyen, Patrick Nguyen, Viet-An Nguyen, Jian-Yun Nie, Rodney Nielsen, Malvina Nissim, Joakim Nivre*, Scott Nowson, Elmar Nöth

Brendan O'Connor*, Tim O'Donnell*, Douglas O'Shaughnessy, Jon Oberlander*, Jan Odijk, Stephan Oepen, Kemal Oflazer, Alice Oh*, Jong-Hoon Oh, Manabu Okumura, Alessandro Oltramari, Vicente Ordonez, Petya Osenova, Cecilia Ovesdotter Alm*, Diarmuid Ó Séaghdha

Ulrike Pado, Martha Palmer, Patrick Pantel*, Antonio Pareja-Lora*, Cecile Paris, Patrick Paroubek, Kristen Parton, Rebecca J. Passonneau, Siddharth Patwardhan, Michael J. Paul*, Adam Pauls*, Adam Pease, Bolette Pedersen, Ted Pedersen, Thomas Pellegrini, Anselmo Peñas, Wim Peters, Slav Petrov, Sasa Petrovic*, Verónica Pérez-Rosas, Maciej Piasecki, Olivier Pietquin, Daniele Pighin*, Manfred Pinkal, Stelios Piperidis, Emily Pitler, Paul Piwek, Barbara Plank, Lonneke

van der Plas*, Massimo Poesio, Simone Paolo Ponzetto, Hoifung Poon, Fred Popowich, Matt Post, Sameer Pradhan, John Prager, Rashmi Prasad, Daniel Preoţiuc-Pietro, Laurent Prévot, Gábor Prószéky, Emily Prud'hommeaux, Adam Przepiórkowski, Stephen Pulman, Matthew Purver, James Pustejovsky, Sampo Pyysalo*

Guojun Qi, Xian Qian, Xipeng Qiu, Ariadna Quattoni, Chris Quirk, Valeria Quochi

Stephan Raaijmakers, Altaf Rahman, Maya Ramanath, Carlos Ramisch, Delip Rao, Sujith Ravi, Marta Recasens, Sravana Reddy, Roi Reichart*, Joseph Reisinger, Ehud Reiter, Norbert Reithinger, David Reitter, Sebastian Riedel, Jason Riesa, Stefan Riezler, German Rigau, Ellen Riloff*, Antonio Roque, Paolo Rosso, Michael Roth, Dominic Rout, Alla Rozovskaya, Frank Rudzicz, Anna Rumshisky, Alexander M. Rush*, Delia Rusu

Markus Saers, Kenji Sagae, Benoît Sagot, Saurav Sahay, Patrick Saint-Dizier, Tetsuya Sakai, Mehdi Samadi, Mark Sammons*, Murat Saraclar, Anoop Sarkar, Felix Sasaki, Giorgio Satta, Roser Saurí, David Schlangen, Helmut Schmid, Nathan Schneider, Hinrich Schuetze, William Schuler, Sabine Schulte im Walde, Roy Schwartz*, Holger Schwenk, Djamé Seddah, Frederique Segond, Yohei Seki, Satoshi Sekine, Jean Senellart, Pavel Serdyukov, Hendra Setiawan, Serge Sharoff, Libin Shen, Wade Shen, Xiaodong Shi, Shuming Shi, Masashi Shimbo, Luke Shrimpton, Advaith Siddharthan, Khalil Sima'an, Michel Simard, Sameer Singh, Jeffrey Mark Siskind, Gabriel Skantze, Kevin Small*, Noah A. Smith*, Nathaniel Smith, Pavel Smrz, Benjamin Snyder, Richard Socher, Thamar Solorio, Swapna Somasundaran, Yang Song, Lucia Specia, Valentin Spitovsky, Caroline Sporleder*, Richard Sproat, Rachele Sprugnoli, Rohini Srihari, Vivek Srikumar, Edward Stabler, Manfred Stede, Armando Stellato*, Amanda Stent, Keith Stevens, Mark Stevenson, Veselin Stoyanov, Carlo Strapparava, Matt Stuttle, Sara Stymne, Keh-Yih Su, Jian Su, Amarnag Subramanya, Fabian Suchanek, David Suendermann, Le Sun, Ang Sun, Ke Sun*, Weiwei Sun, Xu Sun, Mihai Surdeanu, Hisami Suzuki, Jun Suzuki, Stan Szpakowicz*, Idan Szpektor, Anders Søgaard

Whitney Tabor, Marko Tadić, Hiroya Takamura, David Talbot, Partha P. Talukdar, Kumiko Tanaka-Ishii, Oscar Täckström*, Joel Tetreault, Jörg Tiedemann, Christoph Tillmann, Ivan Titov*, Takenobu Tokunaga, Sara Tonelli, Isabel Trancoso, Reut Tsarfaty*, Hajime Tsukada, Yoshimasa Tsuruoka, Dan Tufiș, Gokhan Tur

Olga Uryupina, Jakob Uszkoreit*

Gerhard Van Huyssteen, Keith Vander Linden, Andrejs Vasiļjevs, Paola Velardi, Marc Verhagen*, Yannick Versley, Cristina Vertan, Renata Vieira, Laure Vieu, David Vilar, Aline Villavicencio*, Marta Villegas, Sami Virpioja, Karthik Visweswariah, Andreas Vlachos, Svitlana Volkova*, Piek Vossen

Michael Walsh, Stephen Wan, Xiaojun Wan, Jun Wang, Lu Wang, Rui Wang, Mengqiu Wang*, Bin Wang, Houfeng Wang, Haifeng Wang, Hsin-Min Wang, Wen Wang, Leo Wanner, Yotaro Watanabe, Bonnie Webber, Furu Wei, Gerhard Weikum, David Weir*, Michael White, Rich Wicentowski, Janyce Wiebe, Jason D Williams*, Sandra Williams, Shuly Wintner, Kristian Woodsend*, Dekai Wu, Stephen Wu, Xianchao Wu, Joern Wuebker

Fei Xia, Tong Xiao, Jun Xie, Wei Xu*, Ying Xu, Nianwen Xue, Xiaobing Xue

Bishan Yang, Muyun Yang, Yi Yang, Roman Yangarber, Tae Yano, Limin Yao, Alexander Yates, Mark Yatskar*, Xing Yi, Anssi Yli-Jyrä, Bei Yu, Hong Yu, Nicholas Yuan, François Yvon

David Zajic, Rabih Zbib, Xiaodong Zeng, Richard Zens, Torsten Zesch, Luke Zettlemoyer, Deniz Zeyrek, Ke Zhai, Congle Zhang, Dongdong Zhang, Hao Zhang, Jiajun Zhang, Joy Ying Zhang, Lei Zhang, Qi Zhang, Yue Zhang*, Min Zhang, Yu Zhang, Jian Zhang, Min Zhang, Liu Zhanyi, Jun Zhao, Kai Zhao, Tiejun Zhao, Yanyan Zhao, Guangyou Zhou, Ming Zhou*, Tom Chao Zhou, Qiang Zhou, Jun Zhu, Jingbo Zhu, Chengqing Zong, Ingrid Zukerman, Geoffrey Zweig, Pierre

Zweigenbaum

Secondary Reviewers

Elias Aamot, Azad Abad, Waleed Ammar, Henry Anaya, Mihael Arcan

Timothy Baldwin, Marilena di Bari, Fernando Batista, Lisa Beinborn, David Belanger, Vivek Beniwal, Georgeta Bordea, Djallel Bouneffouf, Hendrik Buschmeier

Hailong Cao, Lin Chen, Wenliang Chen, Bin Chen, Emmanuele Chersoni, Md Faisal Mahbub Chowdhury, Elisabet Comelles Pujadas

Pradipto Das, Li Dong

Nico Erbs

Jeff Flanigan

Dimitrios Galanis, Wei Gao, Diman Ghazi, Roger Granada, Yufan Guo

Masato Hagiwara, Amir Hossein Razavi, Dirk Hovy, Yanzhou Huang

Savvas Karagiannidis, Catherine Kobus, Lingpeng Kong, Florian Kunneman

Patrick Lange, Zhenghua Li, Dingcheng Li, Chen Li, Wang Ling, Daniel Liu, Nikhil Londhe

Zongyang Ma, Ji Ma, Meladel Mistica, Johanna Monti, Masud Moshtaghi

Aida Nematzadeh, Susanne Neumann

Ioannis Pavlopoulos, Daniel Peterson

Steffen Remus, Martin Riedl, Eugen Ruppert

Christer Samuelsson, Germán Sanchis-Trilles, Christophe Servan, Jingsong Su, Elior Sulem

Sam Thomson, Juan-Manuel Torres-Moreno, Jeremy Trione

Yining Wang, Shumin Wu

Deyi Xiong

Feifei Zhai, Chunyue Zhang, Meng Zhang, Kai Zhao

Invited Talk: Learning Ensembles of Structured Prediction Rules

Corinna Cortes
Google Research, New York

Abstract

We present a series of algorithms with theoretical guarantees for learning accurate ensembles of several structured prediction rules for which no prior knowledge is assumed. This includes a number of randomized and deterministic algorithms devised by converting on-line learning algorithms to batch ones, and a boosting-style algorithm applicable in the context of structured prediction with a large number of labels. We also report the results of extensive experiments with these algorithms.

This is joint work with Vitaly Kuznetsov, NYU, and Mehryar Mohri, NYU/Google Research.

Biography

Corinna Cortes is the Head of Google Research, NY, where she is working on a broad range of theoretical and applied large-scale machine learning problems. Prior to Google, Corinna spent more than ten years at AT&T Labs - Research, formerly AT&T Bell Labs, where she held a distinguished research position. Corinna's research work is well-known in particular for her contributions to the theoretical foundations of support vector machines (SVMs), for which she jointly with Vladimir Vapnik received the 2008 Paris Kanellakis Theory and Practice Award, and her work on data-mining in very large data sets for which she was awarded the AT&T Science and Technology Medal in the year 2000. Corinna received her MS degree in Physics from University of Copenhagen and joined AT&T Bell Labs as a researcher in 1989. She received her Ph.D. in computer science from the University of Rochester in 1993. Corinna is also a competitive runner.

Invited Talk: Text Generation for Infinitely Adaptable Curricula

Zoran Popović
Center for Game Science, Computer Science & Engineering, University of Washington

Abstract

Recent studies show that to achieve mastery of a topic by 95% of the student population, some students need ten times more learning content than is available in current curricula. At issue is not just increased volume, but the need for a highly differentiated content specialized to promote optimal learning for each unique learner. To address this synthesis problem we have developed a generative platform capable of dynamically varying content based on the individual student needs. This approach recently achieved 93% mastery of a key algebra concept even for primary school students in three state-wide challenges. In this talk I will describe our work on extending the platform to enable students to solve all word problems in high-school within their preferred context (e.g. sci-fi, medieval, Harry Potter), as well as to automatically generate adaptive learning progressions for reading comprehension curricula in middle school.

Biography

Zoran Popović is a Director of Center for Game Science at University of Washington and founder of Engaged Learning. Trained as a computer scientist his research focus is on creating interactive engaging environments for learning and scientific discovery. His laboratory created Foldit, a biochemistry game that produced three Nature publications in just two years, an award-winning math learning games played by over five million learners worldwide. He is currently focusing on engaging methods that can rapidly develop experts in arbitrary domains with particular focus on revolutionizing K-12 math education. He has recently founded Engaged Learning to apply his work on generative adaptation to any curricula towards the goal of achieving school mastery by 95% of students. His contributions to the field of interactive computer graphics have been recognized by a number of awards including the NSF CAREER Award, Alfred P. Sloan Fellowship and ACM SIGGRAPH Significant New Researcher Award.

Table of Contents

Conference Program

Sunday, June 22, 2014

7:30–18:00 Registration

7:30–9:00 Breakfast

9:00–12:30 Morning Tutorial

Session T1: Gaussian Processes for Natural Language Processing

Session T2: Scalable Large-Margin Structured Learning: Theory and Algorithms

Session T3: Semantics for Large-Scale Multimedia: New Challenges for NLP

Session T4: Wikification and Beyond: The Challenges of Entity and Concept Grounding

12:30–14:00 Lunch break

14:00–17:30 Afternoon Tutorial

Session T5: New Directions in Vector Space Models of Meaning

Session T6: Structured Belief Propagation for NLP

Session T7: Semantics, Discourse and Statistical Machine Translation

Session T8: Syntactic Processing Using Global Discriminative Learning and Beam-Search Decoding

18:00–21:00 Welcome Reception

Monday, June 23, 2014

7:30–18:00 Registration

7:30–9:00 Breakfast

8:55–9:00 Opening session

9:00–9:40 President talk

9:40–10:10 Coffee break

Session 1A: Discourse, Dialogue, Coreference and Pragmatics

10:10–10:35 *Representation Learning for Text-level Discourse Parsing*
Yangfeng Ji and Jacob Eisenstein

10:35–11:00 *Text-level Discourse Dependency Parsing*
Sujian Li, Liang Wang, Ziqiang Cao and Wenjie Li

11:00–11:25 *Discovering Latent Structure in Task-Oriented Dialogues*
Ke Zhai and Jason D Williams

11:25–11:50 *Learning Structured Perceptrons for Coreference Resolution with Latent Antecedents and Non-local Features*
Anders Björkelund and Jonas Kuhn

Monday, June 23, 2014 (continued)

Session 1B: Semantics I

10:10–10:35 *Multilingual Models for Compositional Distributed Semantics*
Karl Moritz Hermann and Phil Blunsom

10:35–11:00 *Simple Negation Scope Resolution through Deep Parsing: A Semantic Solution to a Semantic Problem*
Woodley Packard, Emily M. Bender, Jonathon Read, Stephan Oepen and Rebecca Dridan

11:00–11:25 *Logical Inference on Dependency-based Compositional Semantics*
Ran Tian, Yusuke Miyao and Takuya Matsuzaki

11:25–11:50 *A practical and linguistically-motivated approach to compositional distributional semantics*
Denis Paperno, Nghia The Pham and Marco Baroni

Session 1C: Machine Translation I

10:10–10:35 *Lattice Desegmentation for Statistical Machine Translation*
Mohammad Salameh, Colin Cherry and Grzegorz Kondrak

10:35–11:00 *Bilingually-constrained Phrase Embeddings for Machine Translation*
Jiajun Zhang, Shujie Liu, Mu Li, Ming Zhou and Chengqing Zong

11:00–11:25 *Learning New Semi-Supervised Deep Auto-encoder Features for Statistical Machine Translation*
Shixiang Lu, Zhenbiao Chen and Bo Xu

11:25–11:50 *Learning Topic Representation for SMT with Neural Networks*
Lei Cui, Dongdong Zhang, Shujie Liu, Qiming Chen, Mu Li, Ming Zhou and Muyun Yang

Monday, June 23, 2014 (continued)

Session 1D: Syntax, Parsing and Tagging I

11:00–11:25 *Tagging The Web: Building A Robust Web Tagger with Neural Network*
Ji Ma, Yue Zhang and Jingbo Zhu

Session 1E: NLP for the Web and Social Media I

10:10–10:35 *Unsupervised Solution Post Identification from Discussion Forums*
Deepak P and Karthik Visweswariah

10:35–11:00 *Weakly Supervised User Profile Extraction from Twitter*
Jiwei Li, Alan Ritter and Eduard Hovy

11:00–11:25 *The effect of wording on message propagation: Topic- and author-controlled natural experiments on Twitter*
Chenhao Tan, Lillian Lee and Bo Pang

11:25–11:50 *Inferring User Political Preferences from Streaming Communications*
Svitlana Volkova, Glen Coppersmith and Benjamin Van Durme

11:50–13:20 Lunch break; Student Lunch

Session 2A: Syntax, Parsing and Tagging II

13:20–13:45 *Steps to Excellence: Simple Inference with Refined Scoring of Dependency Trees*
Yuan Zhang, Tao Lei, Regina Barzilay, Tommi Jaakkola and Amir Globerson

13:45–14:10 *Sparser, Better, Faster GPU Parsing*
David Hall, Taylor Berg-Kirkpatrick and Dan Klein

14:10–14:35 *Shift-Reduce CCG Parsing with a Dependency Model*
Wenduan Xu, Stephen Clark and Yue Zhang

14:35–15:00 *Less Grammar, More Features*
David Hall, Greg Durrett and Dan Klein

Monday, June 23, 2014 (continued)

Session 2B: Semantics II

13:20–13:45 *Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors*
Marco Baroni, Georgiana Dinu and Germán Kruszewski

13:45–14:10 *Metaphor Detection with Cross-Lingual Model Transfer*
Yulia Tsvetkov, Leonid Boytsov, Anatole Gershman, Eric Nyberg and Chris Dyer

14:10–14:35 *Learning Word Sense Distributions, Detecting Unattested Senses and Identifying Novel Senses Using Topic Models*
Jey Han Lau, Paul Cook, Diana McCarthy, Spandana Gella and Timothy Baldwin

14:35–15:00 *Learning to Automatically Solve Algebra Word Problems*
Nate Kushman, Luke Zettlemoyer, Regina Barzilay and Yoav Artzi

Session 2C: Word Segmentation and POS Tagging

13:45–14:10 *Modelling function words improves unsupervised word segmentation*
Mark Johnson, Anne Christophe, Emmanuel Dupoux and Katherine Demuth

14:35–15:00 *Max-Margin Tensor Neural Network for Chinese Word Segmentation*
Wenzhe Pei, Tao Ge and Baobao Chang

Session 2D: SRW

Session 2E: Sentiment Analysis I

13:20–13:45 *An Empirical Study on the Effect of Negation Words on Sentiment*
Xiaodan Zhu, Hongyu Guo, Saif Mohammad and Svetlana Kiritchenko

13:45–14:10 *Extracting Opinion Targets and Opinion Words from Online Reviews with Graph Co-ranking*
Kang Liu, Liheng Xu and Jun Zhao

14:10–14:35 *Context-aware Learning for Sentence-level Sentiment Analysis with Posterior Regularization*
Bishan Yang and Claire Cardie

14:35–15:00 *Product Feature Mining: Semantic Clues versus Syntactic Constituents*
Liheng Xu, Kang Liu, Siwei Lai and Jun Zhao

Monday, June 23, 2014 (continued)

15:00–15:30 Coffee break

Session 3A: Topic Modeling

15:30–15:55 *Aspect Extraction with Automated Prior Knowledge Learning*
Zhiyuan Chen, Arjun Mukherjee and Bing Liu

15:55–16:20 *Anchors Regularized: Adding Robustness and Extensibility to Scalable Topic-Modeling Algorithms*
Thang Nguyen, Yuening Hu and Jordan Boyd-Graber

16:20–16:45 *A Bayesian Mixed Effects Model of Literary Character*
David Bamman, Ted Underwood and Noah A. Smith

Session 3B: Information Extraction I

15:30–15:55 *Collective Tweet Wikification based on Semi-supervised Graph Regularization*
Hongzhao Huang, Yunbo Cao, Xiaojiang Huang, Heng Ji and Chin-Yew Lin

15:55–16:20 *Zero-shot Entity Extraction from Web Pages*
Panupong Pasupat and Percy Liang

16:20–16:45 *Incremental Joint Extraction of Entity Mentions and Relations*
Qi Li and Heng Ji

Session 3C: Generation

15:30–15:55 *That's Not What I Meant! Using Parsers to Avoid Structural Ambiguities in Generated Text*
Manjuan Duan and Michael White

15:55–16:20 *Surface Realisation from Knowledge-Bases*
Bikash Gyawali and Claire Gardent

16:20–16:45 *Hybrid Simplification using Deep Semantics and Machine Translation*
Shashi Narayan and Claire Gardent

Monday, June 23, 2014 (continued)

Session 3D: Syntax, Parsing and Tagging III

15:55–16:20 *Grammatical Relations in Chinese: GB-Ground Extraction and Data-Driven Parsing*
Weiwei Sun, Yantao Du, Xin Kou, Shuoyang Ding and Xiaojun Wan

16:20–16:45 *Ambiguity-aware Ensemble Training for Semi-supervised Dependency Parsing*
Zhenghua Li, Min Zhang and Wenliang Chen

Session 3E: Language Resources and Evaluation I

15:30–15:55 *A Robust Approach to Aligning Heterogeneous Lexical Resources*
Mohammad Taher Pilehvar and Roberto Navigli

15:55–16:20 *Predicting the relevance of distributional semantic similarity with contextual information*
Philippe Muller, Cécile Fabre and Clémentine Adam

16:45–17:00 Break

17:00–18:00 Invited talk I: Corinna Cortes

17:00–18:00 *Learning Ensembles of Structured Prediction Rules*
Corinna Cortes, Vitaly Kuznetsov and Mehryar Mohri

Oral Sessions for Student Research Workshop Posters

18:50–21:30 Poster and Dinner Session I: TACL Papers, Long Papers, Short Papers, Student Research Workshop; Demonstrations

Interpretable Semantic Vectors from a Joint Model of Brain- and Text- Based Meaning
Alona Fyshe, Partha P. Talukdar, Brian Murphy and Tom M. Mitchell

Single-Agent vs. Multi-Agent Techniques for Concurrent Reinforcement Learning of Negotiation Dialogue Policies
Kallirroi Georgila, Claire Nelson and David Traum

A Linear-Time Bottom-Up Discourse Parser with Constraints and Post-Editing
Vanessa Wei Feng and Graeme Hirst

Negation Focus Identification with Contextual Discourse Information
Bowei Zou, Guodong Zhou and Qiaoming Zhu

New Word Detection for Sentiment Analysis
Minlie Huang, Borui Ye, Yichen Wang, Haiqiang Chen, Junjun Cheng and Xiaoyan Zhu

ReNew: A Semi-Supervised Framework for Generating Domain-Specific Lexicons and Sentiment Analysis
Zhe Zhang and Munindar P. Singh

A Decision-Theoretic Approach to Natural Language Generation
Nathan McKinley and Soumya Ray

Generating Code-switched Text for Lexical Learning
Igor Labutov and Hod Lipson

Omni-word Feature and Soft Constraint for Chinese Relation Extraction
Yanping Chen, Qinghua Zheng and Wei Zhang

Bilingual Active Learning for Relation Classification via Pseudo Parallel Corpora
Longhua Qian, Haotian Hui, Ya'nan Hu, Guodong Zhou and Qiaoming Zhu

Learning Soft Linear Constraints with Application to Citation Field Extraction
Sam Anzaroot, Alexandre Passos, David Belanger and Andrew McCallum

A Study of Concept-based Weighting Regularization for Medical Records Search
Yue Wang, Xitong Liu and Hui Fang

Learning to Predict Distributions of Words Across Domains
Danushka Bollegala, David Weir and John Carroll

How to make words with vectors: Phrase generation in distributional semantics
Georgiana Dinu and Marco Baroni

Vector space semantics with frequency-driven motifs
Shashank Srivastava and Eduard Hovy

Lexical Inference over Multi-Word Predicates: A Distributional Approach
Omri Abend, Shay B. Cohen and Mark Steedman

A Convolutional Neural Network for Modelling Sentences
Nal Kalchbrenner, Edward Grefenstette and Phil Blunsom

Online Learning in Tensor Space
Yuan Cao and Sanjeev Khudanpur

Graph-based Semi-Supervised Learning of Translation Models from Monolingual Data
Avneesh Saluja, Hany Hassan, Kristina Toutanova and Chris Quirk

Using Discourse Structure Improves Machine Translation Evaluation
Francisco Guzmán, Shafiq Joty, Lluís Màrquez and Preslav Nakov

Learning Continuous Phrase Representations for Translation Modeling
Jianfeng Gao, Xiaodong He, Wen-tau Yih and Li Deng

Adaptive Quality Estimation for Machine Translation
Marco Turchi, Antonios Anastasopoulos, José G. C. de Souza and Matteo Negri

Learning Grounded Meaning Representations with Autoencoders
Carina Silberer and Mirella Lapata

Joint POS Tagging and Transition-based Constituent Parsing in Chinese with Non-local Features
Zhiguo Wang and Nianwen Xue

Strategies for Contiguous Multiword Expression Analysis and Dependency Parsing
Marie Candito and Matthieu Constant

Correcting Preposition Errors in Learner English Using Error Case Frames and Feedback Messages
Ryo Nagata, Mikko Vilenius and Edward Whittaker

Tuesday, June 24, 2014

7:30–18:00 Registration

7:30–9:00 Breakfast

9:00–10:00 Invited talk II: Zoran Popović

9:00–10:00 *Text Generation for Infinitely Adaptable Curricula*
Zoran Popović

10:00–10:30 Coffee break

Session 4A: Machine Learning for NLP

10:30–10:55 *Kneser-Ney Smoothing on Expected Counts*
Hui Zhang and David Chiang

10:55–11:20 *Robust Entity Clustering via Phylogenetic Inference*
Nicholas Andrews, Jason Eisner and Mark Dredze

11:20–11:45 *Linguistic Structured Sparsity in Text Categorization*
Dani Yogatama and Noah A. Smith

11:45–12:10 *Perplexity on Reduced Corpora*
Hayato Kobayashi

Tuesday, June 24, 2014 (continued)

Session 4B: Information Extraction II

10:30–10:55 *Robust Domain Adaptation for Relation Extraction via Clustering Consistency*
Minh Luan Nguyen, Ivor W. Tsang, Kian Ming A. Chai and Hai Leong Chieu

10:55–11:20 *Encoding Relation Requirements for Relation Extraction via Joint Inference*
Liwei Chen, Yansong Feng, Songfang Huang, Yong Qin and Dongyan Zhao

11:20–11:45 *Medical Relation Extraction with Manifold Models*
Chang Wang and James Fan

11:45–12:10 *Distant Supervision for Relation Extraction with Matrix Completion*
Miao Fan, Deli Zhao, Qiang Zhou, Zhiyuan Liu, Thomas Fang Zheng and Edward Y. Chang

Session 4C: Machine Translation II

10:30–10:55 *Enhancing Grammatical Cohesion: Generating Transitional Expressions for SMT*
Mei Tu, Yu Zhou and Chengqing Zong

10:55–11:20 *Adaptive HTER Estimation for Document-Specific MT Post-Editing*
Fei Huang, Jian-Ming Xu, Abraham Ittycheriah and Salim Roukos

11:20–11:45 *Translation Assistance by Translation of L1 Fragments in an L2 Context*
Maarten van Gompel and Antal van den Bosch

11:45–12:10 *Response-based Learning for Grounded Machine Translation*
Stefan Riezler, Patrick Simianer and Carolin Haas

Tuesday, June 24, 2014 (continued)

Session 4D: Summarization

10:30–10:55 *Modelling Events through Memory-based, Open-IE Patterns for Abstractive Summarization*
Daniele Pighin, Marco Cornolti, Enrique Alfonseca and Katja Filippova

10:55–11:20 *Hierarchical Summarization: Scaling Up Multi-Document Summarization*
Janara Christensen, Stephen Soderland, Gagan Bansal and Mausam

11:20–11:45 *Query-Chain Focused Summarization*
Tal Baumel, Raphael Cohen and Michael Elhadad

11:45–12:10 *Exploiting Timelines to Enhance Multi-document Summarization*
Jun-Ping Ng, Yan Chen, Min-Yen Kan and Zhoujun Li

Session 4E: Language Resources and Evaluation II

10:55–11:20 *A chance-corrected measure of inter-annotator agreement for syntax*
Arne Skjærholt

11:20–11:45 *Two Is Bigger (and Better) Than One: the Wikipedia Bitaxonomy Project*
Tiziano Flati, Daniele Vannella, Tommaso Pasini and Roberto Navigli

12:10–13:30 Lunch break

Session 5A: Question Answering

13:30–13:55 *Information Extraction over Structured Data: Question Answering with Freebase*
Xuchen Yao and Benjamin Van Durme

13:55–14:20 *Knowledge-Based Question Answering as Machine Translation*
Junwei Bao, Nan Duan, Ming Zhou and Tiejun Zhao

14:20–14:45 *Discourse Complements Lexical Semantics for Non-factoid Answer Reranking*
Peter Jansen, Mihai Surdeanu and Peter Clark

Tuesday, June 24, 2014 (continued)

Session 5B: Information Extraction III

13:30–13:55 *Toward Future Scenario Generation: Extracting Event Causality Exploiting Semantic Relation, Context, and Association Features*
Chikara Hashimoto, Kentaro Torisawa, Julien Kloetzer, Motoki Sano, István Varga, Jong-Hoon Oh and Yutaka Kidawara

13:55–14:20 *Cross-narrative Temporal Ordering of Medical Events*
Preethi Raghavan, Eric Fosler-Lussier, Noémie Elhadad and Albert M. Lai

Language-Aware Truth Assessment of Fact Candidates
Ndapandula Nakashole and Tom M. Mitchell

Session 5C: Lexical Semantics and Ontology I

13:30–13:55 *That's sick dude!: Automatic identification of word sense change across different timescales*
Sunny Mitra, Ritwik Mitra, Martin Riedl, Chris Biemann, Animesh Mukherjee and Pawan Goyal

13:55–14:20 *A Step-wise Usage-based Method for Inducing Polysemy-aware Verb Classes*
Daisuke Kawahara, Daniel W. Peterson and Martha Palmer

14:20–14:45 *Structured Learning for Taxonomy Induction with Belief Propagation*
Mohit Bansal, David Burkett, Gerard de Melo and Dan Klein

Session 5D: Syntax, Parsing and Tagging IV

13:30–13:55 *A Provably Correct Learning Algorithm for Latent-Variable PCFGs*
Shay B. Cohen and Michael Collins

13:55–14:20 *Spectral Unsupervised Parsing with Additive Tree Metrics*
Ankur P. Parikh, Shay B. Cohen and Eric P. Xing

Tuesday, June 24, 2014 (continued)

Session 5E: Cognitive Modeling and Psycholinguistics

13:30–13:55 *Weak semantic context helps phonetic learning in a model of infant language acquisition*
Stella Frank, Naomi H. Feldman and Sharon Goldwater

13:55–14:20 *Bootstrapping into Filler-Gap: An Acquisition Story*
Marten van Schijndel and Micha Elsner

14:20–14:45 *Nonparametric Learning of Phonological Constraints in Optimality Theory*
Gabriel Doyle, Klinton Bicknell and Roger Levy

14:45–15:15 Coffee break

Session 6A: Machine Translation III

Session 6B: Lexical Semantics and Ontology II

Session 6C: Generation/Summarization/Dialogue

Session 6D: NLP Applications and NLP Enabled Technology I

Session 6E: Language Resources and Evaluation III

16:50–19:20 Poster and Dinner Session II: Long Papers, Short Papers and Demonstrations

Active Learning with Efficient Feature Weighting Methods for Improving Data Quality and Classification Accuracy
Justin Martineau, Lu Chen, Doreen Cheng and Amit Sheth

Political Ideology Detection Using Recursive Neural Networks
Mohit Iyyer, Peter Enns, Jordan Boyd-Graber and Philip Resnik

A Unified Model for Soft Linguistic Reordering Constraints in Statistical Machine Translation
Junhui Li, Yuval Marton, Philip Resnik and Hal Daumé III

Are Two Heads Better than One? Crowdsourced Translation via a Two-Step Collaboration of Non-Professional Translators and Editors
Rui Yan, Mingkun Gao, Ellie Pavlick and Chris Callison-Burch

A Generalized Language Model as the Combination of Skipped n-grams and Modified Kneser Ney Smoothing
Rene Pickhardt, Thomas Gottron, Martin Körner, Paul Georg Wagner, Till Speicher and Steffen Staab

A Semiparametric Gaussian Copula Regression Model for Predicting Financial Risks from Earnings Calls
William Yang Wang and Zhenhao Hua

Polylingual Tree-Based Topic Models for Translation Domain Adaptation
Yuening Hu, Ke Zhai, Vladimir Eidelman and Jordan Boyd-Graber

Low-Resource Semantic Role Labeling
Matthew R. Gormley, Margaret Mitchell, Benjamin Van Durme and Mark Dredze

Joint Syntactic and Semantic Parsing with Combinatory Categorial Grammar
Jayant Krishnamurthy and Tom M. Mitchell

Learning Semantic Hierarchies via Word Embeddings
Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang and Ting Liu

Probabilistic Soft Logic for Semantic Textual Similarity
Islam Beltagy, Katrin Erk and Raymond Mooney

Abstractive Summarization of Spoken and Written Conversations Based on Phrasal Queries
Yashar Mehdad, Giuseppe Carenini and Raymond T. Ng

Comparing Multi-label Classification with Reinforcement Learning for Summarisation of Time-series Data
Dimitra Gkatzia, Helen Hastie and Oliver Lemon

Approximation Strategies for Multi-Structure Sentence Compression
Kapil Thadani

Opinion Mining on YouTube
Aliaksei Severyn, Alessandro Moschitti, Olga Uryupina, Barbara Plank and Katja Filippova

Automatic Keyphrase Extraction: A Survey of the State of the Art
Kazi Saidul Hasan and Vincent Ng

Pattern Dictionary of English Prepositions
Ken Litkowski

Looking at Unbalanced Specialized Comparable Corpora for Bilingual Lexicon Extraction
Emmanuel Morin and Amir Hazem

Validating and Extending Semantic Knowledge Bases using Video Games with a Purpose
Daniele Vannella, David Jurgens, Daniele Scarfini, Domenico Toscani and Roberto Navigli

Shallow Analysis Based Assessment of Syntactic Complexity for Automated Speech Scoring
Suma Bhat, Huichao Xue and Su-Youn Yoon

Can You Repeat That? Using Word Repetition to Improve Spoken Term Detection
Jonathan Wintrode and Sanjeev Khudanpur

Character-Level Chinese Dependency Parsing
Meishan Zhang, Yue Zhang, Wanxiang Che and Ting Liu

Unsupervised Dependency Parsing with Transferring Distribution via Parallel Guidance and Entropy Regularization
Xuezhe Ma and Fei Xia

Unsupervised Morphology-Based Vocabulary Expansion
Mohammad Sadegh Rasooli, Thomas Lippincott, Nizar Habash and Owen Rambow

Toward Better Chinese Word Segmentation for SMT via Bilingual Constraints
Xiaodong Zeng, Lidia S. Chao, Derek F. Wong, Isabel Trancoso and Liang Tian

19:30–22:00 Social at the National Aquarium in Baltimore

Wednesday, June 25, 2014

7:30–18:00 Registration

7:30–9:00 Breakfast

Best paper session

9:00–9:30 *Fast and Robust Neural Network Joint Models for Statistical Machine Translation*
Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz and John Makhoul

9:30–10:00 *Low-Rank Tensors for Scoring Dependency Structures*
Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay and Tommi Jaakkola

10:15–10:45 Coffee break

Session 7A: Multimodal NLP/ Lexical Semantics

11:10–11:35 *CoSimRank: A Flexible & Efficient Graph-Theoretic Similarity Measure*
Sascha Rothe and Hinrich Schütze

11:35–12:00 *Is this a wampimuk? Cross-modal mapping between distributional semantics and the visual world*
Angeliki Lazaridou, Elia Bruni and Marco Baroni

Session 7B: Semantics III

10:45–11:10 *Semantic Parsing via Paraphrasing*
Jonathan Berant and Percy Liang

11:10–11:35 *A Discriminative Graph-Based Parser for the Abstract Meaning Representation*
Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer and Noah A. Smith

11:35–12:00 *Context-dependent Semantic Parsing for Time Expressions*
Kenton Lee, Yoav Artzi, Jesse Dodge and Luke Zettlemoyer

12:00–12:25 *Semantic Frame Identification with Distributed Word Representations*
Karl Moritz Hermann, Dipanjan Das, Jason Weston and Kuzman Ganchev

Wednesday, June 25, 2014 (continued)

Session 7C: Machine Translation IV

10:45–11:10 *A Sense-Based Translation Model for Statistical Machine Translation*
Deyi Xiong and Min Zhang

11:10–11:35 *Recurrent Neural Networks for Word Alignment Model*
Akihiro Tamura, Taro Watanabe and Eiichiro Sumita

11:35–12:00 *A Constrained Viterbi Relaxation for Bidirectional Word Alignment*
Yin-Wen Chang, Alexander M. Rush, John DeNero and Michael Collins

12:00–12:25 *A Recursive Recurrent Neural Network for Statistical Machine Translation*
Shujie Liu, Nan Yang, Mu Li and Ming Zhou

Session 7D: NLP Applications and NLP Enabled Technology II

10:45–11:10 *Predicting Instructor's Intervention in MOOC forums*
Snigdha Chaturvedi, Dan Goldwasser and Hal Daumé III

11:10–11:35 *A Joint Graph Model for Pinyin-to-Chinese Conversion with Typo Correction*
Zhongye Jia and Hai Zhao

11:35–12:00 *Smart Selection*
Patrick Pantel, Michael Gamon and Ariel Fuxman

12:00–12:25 *Modeling Prompt Adherence in Student Essays*
Isaac Persing and Vincent Ng

Wednesday, June 25, 2014 (continued)

Session 7E: Sentiment Analysis II

11:10–11:35 *ConnotationWordNet: Learning Connotation over the Word+Sense Network*
Jun Seok Kang, Song Feng, Leman Akoglu and Yejin Choi

11:35–12:00 *Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification*
Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu and Bing Qin

12:00–12:25 *Towards a General Rule for Identifying Deceptive Opinion Spam*
Jiwei Li, Myle Ott, Claire Cardie and Eduard Hovy

12:25–13:30 Lunch break

13:30–15:00 ACL Business Meeting

Session 8A: NLP for the Web and Social Media II

Session 8B: Semantics/Information Extraction

Session 8C: Machine Translation V

Session 8D: Syntax, Parsing and Tagging V

Session 8E: Multilinguality and Multimodal NLP

16:30–17:00 Coffee break

17:00–18:30 Lifetime Achievement Award

18:30–19:00 Closing Session

Learning Ensembles of Structured Prediction Rules

Corinna Cortes
Google Research
111 8th Avenue,
New York, NY 10011
corinna@google.com

Vitaly Kuznetsov
Courant Institute
251 Mercer Street,
New York, NY 10012
vitaly@cims.nyu.edu

Mehryar Mohri
Courant Institute and Google Research
251 Mercer Street,
New York, NY 10012
mohri@cims.nyu.edu

Abstract

We present a series of algorithms with theoretical guarantees for learning accurate ensembles of several structured prediction rules for which no prior knowledge is assumed. This includes a number of randomized and deterministic algorithms devised by converting on-line learning algorithms to batch ones, and a boosting-style algorithm applicable in the context of structured prediction with a large number of labels. We also report the results of extensive experiments with these algorithms.

1 Introduction

We study the problem of learning accurate ensembles of structured prediction experts. Ensemble methods are widely used in machine learning and have been shown to be often very effective (Breiman, 1996; Freund and Schapire, 1997; Smyth and Wolpert, 1999; MacKay, 1991; Freund et al., 2004). However, ensemble methods and their theory have been developed primarily for binary classification or regression tasks. Their techniques do not readily apply to structured prediction problems. While it is straightforward to combine scalar outputs for a classification or regression problem, it is less clear how to combine structured predictions such as phonemic pronunciation hypotheses, speech recognition lattices, parse trees, or alternative machine translations.

Consider for example the problem of devising an ensemble method for pronunciation, a critical component of modern speech recognition (Ghoshal et al., 2009). Often, several pronunciation models or experts are available for transcribing words into sequences of phonemes. These models may have been derived using other machine learning algorithms or they may be based on carefully hand-crafted rules. In general, none of these pronunciation experts is fully accurate and each expert may be making mistakes at different positions along the output sequence. One can hope that a model that *patches together* the pronunciation of different experts could achieve a superior performance.

Similar ensemble structured prediction problems arise in other tasks, including machine translation, part-of-speech tagging, optical character recognition and computer vision, with structures or substructures varying with each task. We seek to tackle all of these problems simultaneously and consider the general setting where the label or output associated to an input $\mathbf{x} \in \mathcal{X}$ is a structure $\mathbf{y} \in \mathcal{Y}$ that can be decomposed and represented by l substructures $y^1, \ldots, y^l$. For the pronunciation example just discussed, $\mathbf{x}$ is a specific word or word sequence and $\mathbf{y}$ its phonemic transcription. A natural choice for the substructures y^k is then the individual phonemes forming $\mathbf{y}$. Other possible choices include n-grams of consecutive phonemes or more general subsequences.

We will assume that the loss function considered admits an additive decomposition over the substructures, as is common in structured prediction. We also assume access to a set of structured prediction experts $h_1, \ldots, h_p$ that we treat as black boxes. Given an input $\mathbf{x} \in \mathcal{X}$, each expert predicts a structure $h_j(\mathbf{x}) = (h_j^1(\mathbf{x}), \ldots, h_j^l(\mathbf{x}))$. The hypotheses h_j may be the output of a structured prediction algorithm such as Conditional Random Fields (Lafferty et al., 2001), Averaged Perceptron (Collins, 2002), StructSVM (Tsochantaridis et al., 2005), Max Margin Markov Networks (Taskar et al., 2004) or the Regression Technique for Learning Transductions (Cortes et al., 2005), or some other algorithmic or human expert. Given a labeled training sample $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m)$, our objective is to use the predictions of these experts

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1–12,
Baltimore, Maryland, USA, June 23-25 2014.

to form an accurate ensemble.

Variants of the ensemble problem just formulated have been studied in the past in the natural language processing and machine learning literature. One of the most recent, and possibly most relevant studies for sequence data is that of (Nguyen and Guo, 2007), which is based on the forward stepwise selection introduced by (Caruana et al., 2004). However, one disadvantage of this greedy approach is that it can be proven to fail to select an optimal ensemble of experts even in favorable cases where a specialized expert is available for each local prediction (Cortes et al., 2014a). Ensemble methods for structured prediction based on bagging, random forests and random subspaces have also been proposed in (Kocev et al., 2013). One of the limitations of this work is that it is applicable only to a very specific class of tree-based experts introduced in that paper. Similarly, a boosting approach was developed in (Wang et al., 2007) but it applies only to local experts. In the context of natural language processing, a variety of different re-ranking techniques have been proposed for somewhat related problems (Collins and Koo, 2005; Zeman and Žabokrtský, 2005; Sagae and Lavie, 2006; Zhang et al., 2009). But, re-ranking methods do not combine predictions at the level of substructures, thus the final prediction of the ensemble coincides with the prediction made by one of the experts, which can be shown to be suboptimal in many cases. Furthermore, these methods typically assume the use of probabilistic models, which is not a requirement in our learning scenario. Other ensembles of probabilistic models have also been considered in text and speech processing by forming a product of probabilistic models via the intersection of lattices (Mohri et al., 2008), or a straightforward combination of the posteriors from probabilistic grammars trained using EM with different starting points (Petrov, 2010), or some other rather intricate techniques in speech recognition (Fiscus, 1997). Finally, an algorithm of (MacKay, 1997) is another example of an ensemble method for structured prediction though it is not addressing directly the problem we are considering.

Most of the references just mentioned do not give a rigorous theoretical justification for the techniques proposed. We are not aware of any prior theoretical analysis for the ensemble structured prediction problem we consider. Here, we present two families of algorithms for learning ensembles of structured prediction rules that both perform well in practice and enjoy strong theoretical guarantees. In Section 3, we develop ensemble methods based on on-line algorithms. To do so, we extend existing on-line-to-batch conversions to our more general setting. In Section 4, we present a new boosting-style algorithm which is applicable even with a large set of classes as in the problem we consider, and for which we present margin-based learning guarantees. Section 5 reports the results of our extensive experiments.[1]

2 Learning scenario

As in standard supervised learning problems, we assume that the learner receives a training sample $S = ((\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m)) \in \mathcal{X} \times \mathcal{Y}$ of m labeled points drawn i.i.d. according to the some distribution $\mathcal{D}$ used both for training and testing. We also assume that the learner has access to a set of p predictors $h_1, \ldots, h_p$ mapping $\mathcal{X}$ to $\mathcal{Y}$ to devise an accurate ensemble prediction. Thus, for any input $\mathbf{x} \in \mathcal{X}$, he can use the prediction of the p experts $h_1(\mathbf{x}), \ldots, h_p(\mathbf{x})$. No other information is available to the learner about these p experts, in particular the way they have been trained or derived is not known to the learner. But, we will assume that the training sample S is distinct from what may have been used for training the algorithms that generated $h_1(\mathbf{x}), \ldots, h_p(\mathbf{x})$.

To simplify our analysis, we assume that the number of substructures $l \geq 1$ is fixed. This does not cause any loss of generality so long as the maximum number of substructures is bounded, which is the case in all the applications we consider. The quality of the predictions is measured by a loss function $L \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ that can be decomposed as a sum of loss functions $\ell_k \colon \mathcal{Y}_k \to \mathbb{R}_+$ over the substructure sets $\mathcal{Y}_k$, that is, for all $\mathbf{y} = (y^1, \ldots, y^l) \in \mathcal{Y}$ with $y^k \in \mathcal{Y}_k$ and $\mathbf{y}' = (y'^1, \ldots, y'^l) \in \mathcal{Y}$ with $y'^k \in \mathcal{Y}_k$,

$$L(\mathbf{y}, \mathbf{y}') = \sum_{k=1}^{l} \ell_k(y^k, y'^k). \tag{1}$$

We will assume in all that follows that the loss function L is bounded: $L(\mathbf{y}, \mathbf{y}') \leq M$ for all

[1] This paper is a modified version of (Cortes et al., 2014a) to which we refer the reader for the proofs of the theorems stated and a more detailed discussion of our algorithms.

$(\mathbf{y}, \mathbf{y}')$ for some $M > 0$. A prototypical example of such loss functions is the normalized Hamming loss L_{Ham}, which is the fraction of substructures for which two labels $\mathbf{y}$ and $\mathbf{y}'$ disagree, thus in that case $\ell_k(y^k, y'^k) = \frac{1}{l} I_{y^k \neq y'^k}$ and $M = 1$.

3 On-line learning approach

In this section, we present an on-line learning solution to the ensemble structured prediction problem just discussed. We first give a new formulation of the problem as that of on-line learning with expert advice, where the experts correspond to the paths of an acyclic automaton. The on-line algorithm generates at each iteration a distribution over the path-experts. A critical component of our approach consists of using these distributions to define a prediction algorithm with favorable generalization guarantees. This requires an extension of the existing on-line-to-batch conversion techniques to the more general case of combining distributions over path-experts, as opposed to combining single hypotheses.

3.1 Path experts

Each expert h_j induces a set of substructure hypotheses $h_j^1, \ldots, h_j^l$. As already discussed, one particular expert may be better at predicting the kth substructure while some other expert may be more accurate at predicting another substructure. Therefore, it is desirable to combine the substructure predictions of all experts to derive a more accurate prediction. This leads us to considering an acyclic finite automaton G such as that of Figure 1 which admits all possible sequences of substructure hypotheses, or, more generally, a finite automaton such as that of Figure 2 which only allows a subset of these sequences.

An automaton such as G compactly represents a set of *path experts*: each path from the initial vertex 0 to the final vertex l is labeled with a sequence of substructure hypotheses $h_{j_1}^1, \ldots, h_{j_l}^l$ and defines a hypothesis which associates to input $\mathbf{x}$ the output $h_{j_1}^1(\mathbf{x}) \cdots h_{j_l}^l(\mathbf{x})$. We will denote by H the set of all path experts. We also denote by h each path expert defined by $h_{j_1}^1, \ldots, h_{j_l}^l$, with $j_k \in \{1, \ldots, p\}$, and denote by h^k its kth substructure hypothesis $h_{j_k}^k$. Our ensemble structure prediction problem can then be formulated as that of selecting the best path expert (or collection of path experts) in G. Note that, in general, the path expert selected does not coincide with any of the original experts $h_1, \ldots, h_p$.

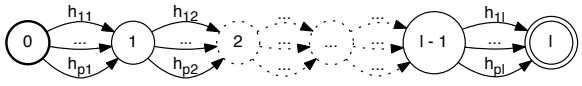


Figure 1: Finite automaton G of path experts.

3.2 On-line algorithm

Using an automaton G, the size of the pool of experts H we consider can be very large. For example, in the case of the automaton of Figure 1, the size of the pool of experts is p^l, and thus is exponentially large with respect to p. But, since learning guarantees in on-line learning admit only a logarithmic dependence on that size, they remain informative in this context. Nevertheless, the computational complexity of most on-line algorithms also directly depends on that size, which could make them impractical in this context. But, there exist several on-line solutions precisely designed to address this issue by exploiting the structure of the experts as in the case of our path experts. These include the algorithm of (Takimoto and Warmuth, 2003) denoted by WMWP, which is an extension of the (randomized) weighted-majority (WM) algorithm of (Littlestone and Warmuth, 1994) to more general bounded loss functions combined with the Weight Pushing (WP) algorithm of (Mohri, 1997); and the Follow the Perturbed Leader (FPL) algorithm of (Kalai and Vempala, 2005). The WMWP algorithm admits a more favorable regret guarantee than the FPL algorithm in our context and our discussion will focus on the use of WMWP for the design of our batch algorithm. However, we have also fully analyzed and implemented a batch algorithm based on FPL (Cortes et al., 2014a).

As in the standard WM algorithm (Littlestone and Warmuth, 1994), WMWP maintains at each round $t \in [1, T]$, a distribution p_t over the set of all experts, which in this context are the path experts $\mathsf{h} \in \mathsf{H}$. At each round $t \in [1, T]$, the algorithm receives an input sequence $\mathbf{x}_t$, incurs the loss $\mathbb{E}_{\mathsf{h} \sim \mathsf{p}_t}[L(\mathsf{h}(\mathbf{x}_t), \mathbf{y}_t)] = \sum_{\mathsf{h}} \mathsf{p}_t(\mathsf{h}) L(\mathsf{h}(\mathbf{x}_t), \mathbf{y}_t)$ and multiplicatively updates the distribution weight per expert:

$$\forall \mathsf{h} \in \mathsf{H}, \mathsf{p}_{t+1}(\mathsf{h}) = \frac{\mathsf{p}_t(\mathsf{h}) \beta^{L(\mathsf{h}(\mathbf{x}_t), \mathbf{y}_t)}}{\sum_{\mathsf{h}' \in \mathsf{H}} \mathsf{p}_t(\mathsf{h}') \beta^{L(\mathsf{h}'(\mathbf{x}_t), \mathbf{y}_t)}}, \quad (2)$$

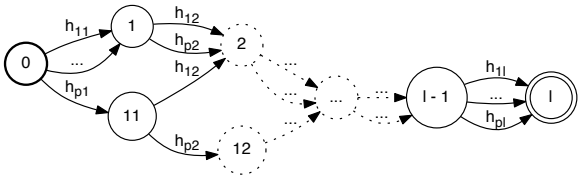


Figure 2: Alternative experts automaton.

where $\beta \in (0,1)$ is some fixed parameter. The number of paths is exponentially large in p and the cost of updating all paths is therefore prohibitive. However, since the loss function is additive in the substructures and the updates are multiplicative, it suffices to maintain instead a weight $w_t(e)$ per transition e, following the update

$$w_{t+1}(e) = \frac{w_t(e)\beta^{\ell_e(\mathbf{x}_t,\mathbf{y}_t)}}{\sum_{orig(e')=\mathrm{orig}(e)} w_t(e')\beta^{\ell_{e'}(\mathbf{x}_t,\mathbf{y}_t)}} \quad (3)$$

where $\ell_e(\mathbf{x}_t, \mathbf{y}_t)$ denotes the loss incurred by the substructure predictor labeling e for the input $\mathbf{x}_t$ and output $\mathbf{y}_t$, and $\mathrm{orig}(e')$ denotes the origin state of a transition e' (Takimoto and Warmuth, 2003). Thus, the cost of the update is then linear in the size of the automaton. To use the resulting weighted automaton for sampling, the weight pushing algorithm is used, whose complexity is also linear in the size of the automaton (Mohri, 1997).

3.3 On-line-to-batch conversion

The WMWP algorithm does not produce a sequence of path experts, rather, a sequence of distributions $\mathsf{p}_1, \ldots, \mathsf{p}_T$ over path experts. Thus, the on-line-to-batch conversion techniques described in (Littlestone, 1989; Cesa-Bianchi et al., 2004; Dekel and Singer, 2005) do not readily apply. Instead, we propose a generalization of the techniques of (Dekel and Singer, 2005). The conversion consists of two steps: extract a good collection of distributions $\mathcal{P} \subseteq \{\mathsf{p}_1, \ldots, \mathsf{p}_T\}$; next use $\mathcal{P}$ to define an accurate hypothesis for prediction. For a subset $\mathcal{P} \subseteq \{\mathsf{p}_1, \ldots, \mathsf{p}_T\}$, we define

$$\Gamma(\mathcal{P}) = \frac{1}{|\mathcal{P}|}\sum_{\mathsf{p}_t \in \mathcal{P}}\sum_{\mathsf{h}\in\mathsf{H}} \mathsf{p}_t(\mathsf{h})L(\mathsf{h}(\mathbf{x}_t),\mathbf{y}_t) + M\sqrt{\frac{\log\frac{1}{\delta}}{|\mathcal{P}|}}$$
$$= \frac{1}{|\mathcal{P}|}\sum_{\mathsf{p}_t \in \mathcal{P}}\sum_{e} w_t(e)\ell_e(\mathbf{x}_t),\mathbf{y}_t) + M\sqrt{\frac{\log\frac{1}{\delta}}{|\mathcal{P}|}},$$

where $\delta > 0$ is a fixed parameter. With this definition, we choose $\mathcal{P}_\delta$ as a minimizer of $\Gamma(\mathcal{P})$ over some collection $\mathcal{P}$ of subsets of $\{\mathsf{p}_1, \ldots, \mathsf{p}_T\}$: $\mathcal{P}_\delta \in \mathrm{argmin}_{\mathcal{P}\in\mathcal{P}}\Gamma(\mathcal{P})$. The choice of $\mathcal{P}$ is restricted by computational considerations. One natural option is to let $\mathcal{P}$ be the union of the suffix sets $\{\mathsf{p}_t, \ldots, \mathsf{p}_T\}$, $t = 1, \ldots, T$. We will assume in what follows that $\mathcal{P}$ includes the set $\{\mathsf{p}_1, \ldots, \mathsf{p}_T\}$.

Next, we define a randomized algorithm based on $\mathcal{P}_\delta$. Given an input $\mathbf{x}$, the algorithm consists of randomly selecting a path h according to

$$\mathsf{p}(\mathsf{h}) = \frac{1}{|\mathcal{P}_\delta|}\sum_{\mathsf{p}_t\in\mathcal{P}_\delta} \mathsf{p}_t(\mathsf{h}), \quad (4)$$

and returning the prediction $\mathsf{h}(\mathbf{x})$. Note that computing and storing p directly is not efficient. To sample from p, we first choose $\mathsf{p}_t \in \mathcal{P}_\delta$ uniformly at random and then sample a path h according to that p_t. Sampling a path according to p_t can be done efficiently using the weight pushing algorithm. Note that once an input $\mathbf{x}$ is received, the distribution p over the path experts h induces a probability distribution $\mathsf{p}_\mathbf{x}$ over the output space $\mathcal{Y}$. It is not hard to see that sampling a prediction $\mathbf{y}$ according to $\mathsf{p}_\mathbf{x}$ is statistically equivalent to first sampling h according to p and then predicting $\mathsf{h}(\mathbf{x})$. We will denote by $\mathcal{H}_{\mathrm{Rand}}$ the randomized hypothesis thereby generated.

An inherent drawback of randomized solutions such as the one just described is that for the same input $\mathbf{x}$ the user can receive different predictions over time. Randomized solutions are also typically more costly to store. A collection of distributions $\mathcal{P}$ can also be used to define a deterministic prediction rule based on the scoring function approach. The majority vote scoring function is defined by

$$\widetilde{\mathsf{h}}_{\mathrm{MVote}}(\mathbf{x},\mathbf{y}) = \prod_{k=1}^{l}\Big(\frac{1}{|\mathcal{P}_\delta|}\sum_{\mathsf{p}_t\in\mathcal{P}_\delta}\sum_{j=1}^{p} w_{t,kj}\mathbf{1}_{h_j^k(\mathbf{x})=y^k}\Big). \quad (5)$$

The majority vote algorithm denoted by $\mathcal{H}_{\mathrm{MVote}}$ is then defined for all $\mathbf{x} \in \mathcal{X}$, by $\mathcal{H}_{\mathrm{MVote}}(\mathbf{x}) = \mathrm{argmax}_{\mathbf{y}\in\mathcal{Y}}\widetilde{\mathsf{h}}_{\mathrm{MVote}}(\mathbf{x},\mathbf{y})$. For an expert automaton accepting all path experts such as that of Figure 1, the maximizer of $\widetilde{\mathsf{h}}_{\mathrm{MVote}}$ can be found very efficiently by choosing $\mathbf{y}$ such that y^k has the maximum weight in position k.

In the next section, we present learning guarantees for $\mathcal{H}_{\mathrm{Rand}}$ and $\mathcal{H}_{\mathrm{MVote}}$. For a more extensive dis-

cussion of alternative prediction rules, see (Cortes et al., 2014a).

3.4 Batch learning guarantees

We first present learning bounds for the randomized prediction rule $\mathcal{H}_{\text{Rand}}$. Next, we upper bound the generalization error of $\mathcal{H}_{\text{MVote}}$ in terms of that of $\mathcal{H}_{\text{Rand}}$.

Theorem 1. *For any $\delta > 0$, with probability at least $1-\delta$ over the choice of the sample $((\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_T, \mathbf{y}_T))$ drawn i.i.d. according to $\mathcal{D}$, the following inequalities hold:*

$$\mathbb{E}[L(\mathcal{H}_{\text{Rand}}(\mathbf{x}), \mathbf{y})] \leq \inf_{\mathsf{h} \in \mathsf{H}} \mathbb{E}[L(\mathsf{h}(\mathbf{x}), \mathbf{y})] + 2M\sqrt{\frac{l \log p}{T}} + 2M\sqrt{\frac{\log \frac{2}{\delta}}{T}}.$$

For the normalized Hamming loss L_{Ham}, the bound of Theorem 1 holds with $M = 1$.

We now upper bound the generalization error of the majority-vote algorithm $\mathcal{H}_{\text{MVote}}$ in terms of that of the randomized algorithm $\mathcal{H}_{\text{Rand}}$, which, combined with Theorem 1, immediately yields generalization bounds for the majority-vote algorithm $\mathcal{H}_{\text{MVote}}$.

Proposition 2. *The following inequality relates the generalization error of the majority-vote algorithm to that of the randomized one:*

$$\mathbb{E}[L_{\text{Ham}}(\mathcal{H}_{\text{MVote}}(\mathbf{x}), \mathbf{y})] \leq 2\,\mathbb{E}[L_{\text{Ham}}(\mathcal{H}_{\text{Rand}}(\mathbf{x}), \mathbf{y})],$$

where the expectations are taken over $(\mathbf{x}, \mathbf{y}) \sim \mathcal{D}$ and $\mathsf{h} \sim \mathsf{p}$.

Proposition 2 suggests that the price to pay for derandomization is a factor of 2. More refined and more favorable guarantees can be proven for the majority-vote algorithm (Cortes et al., 2014a).

4 Boosting-style algorithm

In this section, we devise a boosting-style algorithm for our ensemble structured prediction problem. The variants of AdaBoost for multi-class classification such as AdaBoost.MH or AdaBoost.MR (Freund and Schapire, 1997; Schapire and Singer, 1999; Schapire and Singer, 2000) cannot be readily applied in this context. First, the number of classes to consider here is quite large, as in all structured prediction problems, since it is exponential in the number of substructures l. For example, in the case of the pronunciation problem where the number of phonemes for English is in the order of 50, the number of classes is 50^l. But, the objective function for AdaBoost.MH or AdaBoost.MR as well as the main steps of the algorithms include a sum over all possible labels, whose computational cost in this context would be prohibitive. Second, the loss function we consider is the normalized Hamming loss over the substructures predictions, which does not match the multi-class losses for the variants of AdaBoost.[2] Finally, the natural base hypotheses for this problem admit a structure that can be exploited to devise a more efficient solution, which of course was not part of the original considerations for the design of these variants of AdaBoost.

4.1 Hypothesis sets

The predictor $\mathcal{H}_{\text{Boost}}$ returned by our boosting algorithm is based on a scoring function $\widetilde{\mathsf{h}}\colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, which, as for standard ensemble algorithms such as AdaBoost, is a convex combination of base scoring functions $\widetilde{\mathsf{h}}_t$: $\widetilde{\mathsf{h}} = \sum_{t=1}^{T} \alpha_t \widetilde{\mathsf{h}}_t$, with $\alpha_t \geq 0$. The base scoring functions used in our algorithm have the form

$$\forall (\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}, \quad \widetilde{\mathsf{h}}_t(\mathbf{x}, \mathbf{y}) = \sum_{k=1}^{l} \widetilde{\mathsf{h}}_t^k(\mathbf{x}, \mathbf{y}).$$

In particular, these can be derived from the path experts in H by letting $\mathsf{h}_t^k(\mathbf{x}, \mathbf{y}) = \mathbf{1}_{\mathsf{h}_t^k(\mathbf{x}) = y^k}$. Thus, the score assigned to $\mathbf{y}$ by the base scoring function $\widetilde{\mathsf{h}}_t$ is the number of positions at which $\mathbf{y}$ matches the prediction of path expert h_t given input $\mathbf{x}$. $\mathcal{H}_{\text{Boost}}$ is defined as follows in terms of $\widetilde{\mathsf{h}}$ or h_ts:

$$\forall \mathbf{x} \in \mathcal{X}, \mathcal{H}_{\text{Boost}}(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \widetilde{\mathsf{h}}(\mathbf{x}, \mathbf{y})$$

We remark that the analysis and algorithm presented in this section are also applicable with a scoring function that is the product of the scores

[2] (Schapire and Singer, 1999) also present an algorithm using the Hamming loss for multi-class classification, but that is a Hamming loss over the set of classes and differs from the loss function relevant to our problem. Additionally, the main steps of that algorithm are also based on a sum over all classes.

at each substructure k as opposed to a sum, that is,

$$\widetilde{\mathsf{h}}(\mathbf{x}, \mathbf{y}) = \prod_{k=1}^{l} \left(\sum_{t=1}^{T} \alpha_t \widetilde{\mathsf{h}}_t^k(\mathbf{x}, \mathbf{y}) \right).$$

This can be used for example in the case where the experts are derived from probabilistic models.

4.2 ESPBoost algorithm

To simplify our exposition, the algorithm that we now present uses base learners of the form $\mathsf{h}_t^k(\mathbf{x}, \mathbf{y}) = \mathbf{1}_{\mathsf{h}_t^k(\mathbf{x}) = y^k}$. The general case can be handled in the same fashion with the only difference being the definition of the direction and step of the optimization procedure described below. For any $i \in [1, m]$ and $k \in [1, l]$, we define the *margin of* $\widetilde{\mathsf{h}}^k$ *for point* $(\mathbf{x}_i, \mathbf{y}_i)$ by $\rho(\widetilde{\mathsf{h}}^k, \mathbf{x}_i, \mathbf{y}_i) = \widetilde{\mathsf{h}}^k(\mathbf{x}_i, y_i^k) - \max_{y^k \neq y_i^k} \widetilde{\mathsf{h}}^k(\mathbf{x}_i, y^k)$. We first derive an upper bound on the empirical normalized Hamming loss of a hypothesis $\mathcal{H}_{\text{Boost}}$, with $\widetilde{\mathsf{h}} = \sum_{t=1}^{T} \alpha_t \widetilde{\mathsf{h}}_t$.

Lemma 3. *The following upper bound holds for the empirical normalized Hamming loss of the hypothesis* $\mathcal{H}_{\text{Boost}}$*:*

$$\mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})\sim S}[L_{\text{Ham}}(\mathcal{H}_{\text{Boost}}(\mathbf{x}), \mathbf{y})] \leq \frac{1}{ml} \sum_{i=1}^{m} \sum_{k=1}^{l} \exp\Big(- \sum_{t=1}^{T} \alpha_t \rho(\widetilde{\mathsf{h}}_t^k, \mathbf{x}_i, \mathbf{y}_i) \Big).$$

The proof of this lemma as well as that of several other theorems related to this algorithm can be found in (Cortes et al., 2014a).

In view of this upper bound, we consider the objective function $F\colon \mathbb{R}^N \to \mathbb{R}$ defined for all $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N) \in \mathbb{R}^N$ by

$$F(\boldsymbol{\alpha}) = \frac{1}{ml} \sum_{i=1}^{m} \sum_{k=1}^{l} \exp\Big(- \sum_{j=1}^{N} \alpha_j \rho(\widetilde{\mathsf{h}}_j^k, \mathbf{x}_i, \mathbf{y}_i) \Big),$$

where $\mathsf{h}_1, \ldots, \mathsf{h}_N$ denote the set of all path experts in H. F is a convex and differentiable function of $\boldsymbol{\alpha}$. Our algorithm, ESPBoost (Ensemble Structured Prediction Boosting), is defined by the application of coordinate descent to the objective F. Algorithm 1 shows the pseudocode of the ESPBoost.

Algorithm 1 ESPBoost Algorithm

Inputs: $S = ((\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m))$; set of experts $\{h_1, \ldots, h_p\}$
for $i = 1$ **to** m **and** $k = 1$ **to** l **do**
  $\mathcal{D}_1(i, k) \leftarrow \frac{1}{ml}$
end for
for $t = 1$ **to** T **do**
  $\mathsf{h}_t \leftarrow \operatorname{argmin}_{\mathsf{h} \in \mathsf{H}} \mathbb{E}_{(i,k)\sim \mathcal{D}_t}[\mathbf{1}_{\mathsf{h}^k(\mathbf{x}_i) \neq y_i^k}]$
  $\epsilon_t \leftarrow \mathbb{E}_{(i,k)\sim \mathcal{D}_t}[\mathbf{1}_{\mathsf{h}_t^k(\mathbf{x}_i) \neq y_i^k}]$
  $\alpha_t \leftarrow \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
  $Z_t \leftarrow 2\sqrt{\epsilon_t(1-\epsilon_t)}$
  for $i = 1$ **to** m **and** $k = 1$ **to** l **do**
    $\mathcal{D}_{t+1}(i, k) \leftarrow \frac{\exp(-\alpha_t \rho(\widetilde{\mathsf{h}}_t^k, \mathbf{x}_i, \mathbf{y}_i)) \mathcal{D}_t(i,k)}{Z_t}$
  end for
end for
Return $\widetilde{\mathsf{h}} = \sum_{t=1}^{T} \alpha_t \widetilde{\mathsf{h}}_t$

Let $\boldsymbol{\alpha}_{t-1} \in \mathbb{R}^N$ denote the vector obtained after $t-1$ iterations and $\mathbf{e}_t$ the tth unit vector in $\mathbb{R}^N$. We denote by $\mathcal{D}_t$ the distribution over $[1, m] \times [1, l]$ defined by

$$\mathcal{D}_t(i, k) = \frac{\frac{1}{ml} \exp\Big(- \sum_{u=1}^{t-1} \alpha_u \rho(\widetilde{\mathsf{h}}_u^k, \mathbf{x}_i, \mathbf{y}_i) \Big)}{A_{t-1}}$$

where A_{t-1} is a normalization factor, $A_{t-1} = \frac{1}{ml} \sum_{i=1}^{m} \sum_{k=1}^{l} \exp\big(- \sum_{u=1}^{t-1} \alpha_u \rho(\widetilde{\mathsf{h}}_u^k, \mathbf{x}_i, \mathbf{y}_i) \big)$. The direction $\mathbf{e}_t$ selected at the tth round is the one minimizing the directional derivative, that is

$$\begin{aligned} &\frac{dF(\boldsymbol{\alpha}_{t-1} + \eta \mathbf{e}_t)}{d\eta}\bigg|_{\eta=0} \\ &= - \sum_{i=1}^{m} \sum_{k=1}^{l} \rho(\widetilde{\mathsf{h}}_t^k, \mathbf{x}_i, \mathbf{y}_i) \mathcal{D}_t(i, k) A_{t-1} \\ &= \Big[2 \sum_{i,k:\mathsf{h}_t^k(\mathbf{x}_i) \neq y_i^k} \mathcal{D}_t(i, k) - 1 \Big] A_{t-1} \\ &= (2\epsilon_t - 1) A_{t-1}, \end{aligned}$$

where ϵ_t is the average error of h_t given by

$$\begin{aligned} \epsilon_t &= \sum_{i=1}^{m} \sum_{k=1}^{l} \mathcal{D}_t(i, k) \mathbf{1}_{\mathsf{h}_t^k(\mathbf{x}_i) \neq y_i^k} \\ &= \mathop{\mathbb{E}}_{(i,k)\sim \mathcal{D}_t} [\mathbf{1}_{\mathsf{h}_t^k(\mathbf{x}_i) \neq y_i^k}]. \end{aligned}$$

The remaining steps of our algorithm can be determined as in the case of AdaBoost. In particular, given the direction $\mathbf{e}_t$, the best step α_t is obtained by solving the equation $\frac{dF(\boldsymbol{\alpha}_{t-1} + \alpha_t \mathbf{e}_t)}{d\alpha_t} =$

0, which admits the closed-form solution $\alpha_t = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}$. The distribution $\mathcal{D}_{t+1}$ can be expressed in terms of $\mathcal{D}_t$ with the normalization factor $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$.

Our *weak learning assumption* in this context is that there exists $\gamma > 0$ such that at each round, ϵ_t verifies $\epsilon_t < \frac{1}{2} - \gamma$. Note that, at each round, the path expert h_t with the smallest error ϵ_t can be determined easily and efficiently by first finding for each substructure k, the h_t^k that is the best with respect to the distribution weights $\mathcal{D}_t(i,k)$.

Observe that, while the steps of our algorithm are syntactically close to those of AdaBoost and its multi-class variants, our algorithm is distinct and does not require sums over the exponential number of all possible labelings of the substructures and is quite efficient.

4.3 Learning guarantees

We have derived both a margin-based generalization bound in support of the ESPBoost algorithm and a bound on the empirical margin loss.

For any $\rho > 0$, define the empirical margin loss of $\mathcal{H}_{\text{Boost}}$ by the following:

$$\widehat{R}_\rho\left(\frac{\widetilde{\mathsf{h}}}{\|\boldsymbol{\alpha}\|_1}\right) = \frac{1}{ml}\sum_{i=1}^{m}\sum_{k=1}^{l}\mathbf{1}_{\rho(\widetilde{\mathsf{h}}^k,\mathbf{x}_i,\mathbf{y}_i)\leq\rho\|\boldsymbol{\alpha}\|_1},$$

where $\widetilde{\mathsf{h}}$ is the corresponding scoring function. The following theorem can be proven using the multi-class classification bounds of (Koltchinskii and Panchenko, 2002; Mohri et al., 2012) as can be shown in (Cortes et al., 2014a).

Theorem 4. *Let $\mathcal{F}$ denote the set of functions $\mathcal{H}_{\text{Boost}}$ with $\widetilde{\mathsf{h}} = \sum_{t=1}^{T}\alpha_t\mathsf{h}_t$ for some $\alpha_1,\ldots,\alpha_t \geq 0$ and $\mathsf{h}_t \in \mathsf{H}$ for all $t \in [1,T]$. Fix $\rho > 0$. Then, for any $\delta > 0$, with probability at least $1-\delta$, the following holds for all $\mathcal{H}_{\text{Boost}} \in \mathcal{F}$:*

$$\mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}[L_{\text{Ham}}(\mathcal{H}_{\text{Boost}}(\mathbf{x}),\mathbf{y})] \leq \widehat{R}_\rho\left(\frac{\widetilde{\mathsf{h}}}{\|\boldsymbol{\alpha}\|_1}\right) + \frac{2}{\rho l}\sum_{k=1}^{l}|\mathcal{Y}_k|^2\mathfrak{R}_m(H^k) + \sqrt{\frac{\log\frac{l}{\delta}}{2m}},$$

where $\mathfrak{R}_m(H^k)$ denotes the Rademacher complexity of the class of functions

$$H^k = \{\mathbf{x}\mapsto\widetilde{\mathsf{h}}_t^k : j\in[1,p], y\in\mathcal{Y}_k\}.$$

Table 1: Average Normalized Hamming Loss, ADS1 and ADS2. $\beta_{ADS1} = 0.95$, $\beta_{ADS2} = 0.95$, $T_{SLE} = 100$, $\delta = 0.05$.

	ADS1, $m = 200$	ADS2, $m = 200$
$\mathcal{H}_{\text{MVote}}$	**0.0197 ± 0.00002**	**0.2172 ± 0.00983**
$\mathcal{H}_{\text{FPL}}$	0.0228 ± 0.00947	0.2517 ± 0.05322
$\mathcal{H}_{\text{CV}}$	**0.0197 ± 0.00002**	0.2385 ± 0.00002
$\mathcal{H}_{\text{FPL-CV}}$	0.0741 ± 0.04087	0.4001 ± 0.00028
$\mathcal{H}_{\text{ESPBoost}}$	**0.0197 ± 0.00002**	**0.2267 ± 0.00834**
$\mathcal{H}_{\text{SLE}}$	0.5641 ± 0.00044	0.2500 ± 0.05003
$\mathcal{H}_{\text{Rand}}$	0.1112 ± 0.00540	0.4000 ± 0.00018
Best h_j	0.5635 ± 0.00004	0.4000

This theorem provides a margin-based guarantee for convex ensembles such as those returned by ESPBoost. The following theorem further provides an upper bound on the empirical margin loss for ESPBoost.

Theorem 5. *Let $\widetilde{\mathsf{h}}$ denote the scoring function returned by ESPBoost after $T \geq 1$ rounds. Then, for any $\rho > 0$, the following inequality holds:*

$$\widehat{R}_\rho\left(\frac{\widetilde{\mathsf{h}}}{\|\boldsymbol{\alpha}\|_1}\right) \leq 2^T\prod_{t=1}^{T}\sqrt{\epsilon_t^{1-\rho}(1-\epsilon_t)^{1+\rho}}.$$

As in the case of AdaBoost (Schapire et al., 1997), it can be shown that for $\rho < \gamma$, $\epsilon_t^{1-\rho}(1-\epsilon_t)^{1+\rho} \leq (1-2\gamma)^{1-\rho}(1+2\gamma)^{1+\rho} < 1$ and the right-hand side of this bound decreases exponentially with T.

5 Experiments

We used a number of artificial and real-world data sets for our experiments. For each data set, we performed 10-fold cross-validation with disjoint training sets.[3] We report the average error for each task. In addition to the $\mathcal{H}_{\text{MVote}}$, $\mathcal{H}_{\text{Rand}}$ and $\mathcal{H}_{\text{ESPBoost}}$ hypotheses, we experimented with two algorithms discussed in more detail in (Cortes et al., 2014a): a cross-validation on-line-to-batch conversion of the WMWP algorithm, $\mathcal{H}_{\text{CV}}$, a majority-vote on-line-to-batch conversion with FPL, $\mathcal{H}_{\text{FPL}}$, and a cross-validation on-line-to-batch conversion with FPL, $\mathcal{H}_{\text{FPL-CV}}$. Finally, we compare with the $\mathcal{H}_{\text{SLE}}$ algorithm of (Nguyen and Guo, 2007).

5.1 Artificial data sets

Our artificial data set, ADS1 and ADS2 simulate the scenarios described in Section 1. In ADS1 the

[3] For the OCR data set, these subsets are predefined.

kth expert has a high accuracy on the kth position, in ADS2 an expert has low accuracy in a fixed set of positions.

For the first artificial data set, ADS1, we used local experts $h_1, \ldots, h_p$ with $p = 5$. To generate the data we chose an arbitrary Markov chain over the English alphabet and sampled $40{,}000$ random sequences each consisting of 10 symbols. Each of the five experts was designed to have a certain probability of making a mistake at each position in the sequence. Expert h_j correctly predicted positions $2j-1$ and $2j$ with probability 0.97 and other positions with probability 0.5. We forced experts to make similar mistakes by making them select an adjacent alphabet symbol in case of an error. For example, when a mistake was made on a symbol b, the expert prediction was forced to be either a or c. The second artificial data set, ADS2, modeled the case of rather poor experts. ADS2 was generated in the same way as ADS1, but the expert predictions were different. This time each expert made mistakes at four out of the ten distinct random positions in each sequence.

Table 1 reports the results of our experiments. For all experiments with the algorithms $\mathcal{H}_{\text{Rand}}$, $\mathcal{H}_{\text{MVote}}$, and $\mathcal{H}_{\text{CV}}$, we ran the WMWP algorithm for $T = m$ rounds with the β values listed in the caption of Table 1, generating distributions $\mathcal{P} \subseteq \{\mathsf{p}_1, \ldots, \mathsf{p}_T\}$. For $\mathcal{P}$ we used the collection of all suffix sets $\{\mathsf{p}_t, \ldots, \mathsf{p}_T\}$ and $\delta = 0.05$. For the algorithms based on FPL, we used $\epsilon = 0.5/pl$. The same parameter choices were used for the subsequent experiments.

As can be seen from Table 1, in both cases, $\mathcal{H}_{\text{MVote}}$, our majority-vote algorithm based on our on-line-to-batch conversion using the WMWP algorithm (together with most of the other on-line based algorithms), yields a significant improvement over the best expert. It also outperforms $\mathcal{H}_{\text{SLE}}$, which in the case of ADS1 even fails to outperform the best h_j. After 100 iterations on ADS1, the ensemble learned by $\mathcal{H}_{\text{SLE}}$ consists of a single expert, which is why it leads to such a poor performance.

It is also worth pointing out that $\mathcal{H}_{\text{FPL-CV}}$ and $\mathcal{H}_{\text{Rand}}$ fail to outperform the best model on ADS2 set. This is in total agreement with our theoretical analysis since, in this case, any path expert has exactly the same performance and the error of the best path expert is an asymptotic upper bound on the errors of these algorithms. The superior performance of the majority-vote-based algorithms suggests that these algorithms may have an advantage over other prediction rules beyond what is suggested by our learning bounds.

Table 2: Average Normalized Hamming Loss for ADS3. $\beta_{ADS1} = 0.95$, $\beta_{ADS2} = 0.95$, $T_{SLE} = 100$, $\delta = 0.05$.

$\mathcal{H}_{\text{MVote}}$	$\mathbf{0.1788 \pm 0.00004}$
$\mathcal{H}_{\text{FPL}}$	0.2189 ± 0.04097
$\mathcal{H}_{\text{CV}}$	$\mathbf{0.1788 \pm 0.00004}$
$\mathcal{H}_{\text{FPL-CV}}$	0.3148 ± 0.00387
$\mathcal{H}_{\text{ESPBoost}}$	0.1831 ± 0.00240
$\mathcal{H}_{\text{SLE}}$	0.1954 ± 0.00185
$\mathcal{H}_{\text{Rand}}$	0.3196 ± 0.00018
Best h_j	0.2957 ± 0.00005

Table 3: Average Normalized Hamming Loss, PDS1 and PDS2. $\beta_{PDS1} = 0.85$, $\beta_{PDS2} = 0.97$, $T_{SLE} = 100$, $\delta = 0.05$.

	PDS1, $m = 130$	PDS2, $m = 400$
$\mathcal{H}_{\text{MVote}}$	$\mathbf{0.2225 \pm 0.00301}$	$\mathbf{0.2323 \pm 0.00069}$
$\mathcal{H}_{\text{FPL}}$	0.2657 ± 0.07947	0.2337 ± 0.00229
$\mathcal{H}_{\text{CV}}$	$\mathbf{0.2316 \pm 0.00189}$	$\mathbf{0.2364 \pm 0.00080}$
$\mathcal{H}_{\text{FPL-CV}}$	0.4451 ± 0.02743	0.4090 ± 0.01388
$\mathcal{H}_{\text{ESPBoost}}$	0.3625 ± 0.01054	0.3499 ± 0.00509
$\mathcal{H}_{\text{SLE}}$	0.3130 ± 0.05137	0.3308 ± 0.03182
$\mathcal{H}_{\text{Rand}}$	0.4713 ± 0.00360	0.4607 ± 0.00131
Best h_j	0.3449 ± 0.00368	0.3413 ± 0.00067

We also synthesized a third data set, ADS3. Here, we simulated the case where each expert specialized in predicting some subset of the labels. In particular, we generated $40{,}000$ random sequences over the English alphabet in the same way as for ADS1 and ADS2. To generate expert predictions, we partitioned the alphabet into 5 disjoint subsets A_j. Expert j always correctly predicted the label in A_j and the probability of correctly predicting the label not in A_j was set to 0.7. To train the ensemble algorithms, we used a training set of size $m = 200$.

The results are presented in Table 2. $\mathcal{H}_{\text{MVote}}$, $\mathcal{H}_{\text{CV}}$ and $\mathcal{H}_{\text{ESPBoost}}$ achieve the best performance on this data set with a considerable improvement in accuracy over the best expert h_j. We also observe as for the ADS2 experiment that $\mathcal{H}_{\text{Rand}}$ and $\mathcal{H}_{\text{FPL-CV}}$ fail to outperform the best model and approach the accuracy of the best path expert only asymptotically.

Table 4: Average edit distance, PDS1 and PDS2. $\beta_{PDS1} = 0.85$, $\beta_{PDS2} = 0.97$, $T_{SLE} = 100$, $\delta = 0.05$.

	PDS1, $m = 130$	PDS2, $m = 400$
$\mathcal{H}_{\text{MVote}}$	**0.8395 ± 0.01076**	**0.9626 ± 0.00341**
$\mathcal{H}_{\text{FPL}}$	1.0158 ± 0.34379	0.9744 ± 0.01277
$\mathcal{H}_{\text{CV}}$	**0.8668 ± 0.00553**	**0.9840 ± 0.00364**
$\mathcal{H}_{\text{FPL-CV}}$	1.8044 ± 0.09315	1.8625 ± 0.06016
$\mathcal{H}_{\text{ESPBoost}}$	1.3977 ± 0.06017	1.4092 ± 0.04352
$\mathcal{H}_{\text{SLE}}$	1.1762 ± 0.12530	1.2477 ± 0.12267
$\mathcal{H}_{\text{Rand}}$	1.8962 ± 0.01064	2.0838 ± 0.00518
Best h_j	1.2163 ± 0.00619	1.2883 ± 0.00219

5.2 Pronunciation data sets

We had access to two proprietary pronunciation data sets, PDS1 and PDS2. In both sets, each example is an English word, typically a proper name. For each word, 20 possible phonemic sequences are available, ranked by some pronunciation model. Since the true pronunciation was not available, we set the top sequence to be the target label and used the remaining as the predictions made by the experts. The only difference between PDS1 and PDS2 is their size: 1,313 words for PDS1 and 6,354 for PDS2.

In both cases, on-line based algorithms, specifically $\mathcal{H}_{\text{MVote}}$, significantly outperform the best model as well as $\mathcal{H}_{\text{SLE}}$, see Table 3. The poor performance of $\mathcal{H}_{\text{ESPBoost}}$ is due to the fact that the weak learning assumption is violated after 5-8 iterations and hence the algorithm terminates.

It can be argued that for this task the edit-distance is a more suitable measure of performance than the average Hamming loss. Thus, we also report the results of our experiments in terms of the edit-distance in Table 4. Remarkably, our on-line based algorithms achieve a comparable improvement over the performance of the best model in the case of edit-distance as well.

5.3 OCR data set

Rob Kassel's OCR data set is available for download from http://ai.stanford.edu/~btaskar/ocr/. It contains 6,877 word instances with a total of 52,152 characters. Each character is represented by $16 \times 8 = 128$ binary pixels. The task is to predict a word given its sequence of pixel vectors. To generate experts, we used several software packages: CRFsuite (Okazaki, 2007) and SVM^{struct}, SVM^{multiclass} (Joachims, 2008), and the Stanford Classifier (Rafferty et al., 2014). We trained these algorithms on each of the predefined folds of the data set and generated predictions on the test fold using the resulting models.

Table 5: Average Normalized Hamming Loss, TR1 and TR2. $\beta_{TR1} = 0.95$, $\beta_{TR2} = 0.98$, $T_{SLE} = 100$, $\delta = 0.05$.

	TR1, $m = 800$	TR2, $m = 1000$
$\mathcal{H}_{\text{MVote}}$	**0.0850 ± 0.00096**	**0.0746 ± 0.00014**
$\mathcal{H}_{\text{FPL}}$	**0.0859 ± 0.00110**	**0.0769 ± 0.00218**
$\mathcal{H}_{\text{CV}}$	**0.0843 ± 0.00006**	**0.0741 ± 0.00011**
$\mathcal{H}_{\text{FPL-CV}}$	0.1093 ± 0.00129	0.1550 ± 0.00182
$\mathcal{H}_{\text{ESPBoost}}$	0.1041 ± 0.00056	0.1414 ± 0.00233
$\mathcal{H}_{\text{SLE}}$	**0.0778 ± 0.00934**	**0.0814 ± 0.02558**
$\mathcal{H}_{\text{Rand}}$	0.1128 ± 0.00048	0.1652 ± 0.00077
Best h_j	0.1032 ± 0.00007	0.1415 ± 0.00005

Our results (see (Cortes et al., 2014a)) show that ensemble methods lead only to a small improvement in performance over the best h_j. This is because here the best model h_j dominates all other experts and ensemble methods cannot benefit from patching together different outputs.

5.4 Penn Treebank data set

The part-of-speech task, POS, consists of labeling each word of a sentence with its correct part-of-speech tag. The Penn Treebank 2 data set is available through LDC license at http://www.cis.upenn.edu/~treebank/ and contains 251,854 sentences with a total of 6,080,493 tokens and 45 different parts-of-speech.

For the first experiment, TR1, we used 4 disjoint training sets to produce 4 SVM^{multiclass} models and 4 maximum entropy models using the Stanford Classifier. We also used the union of these training sets to devise one CRFsuite model. For the second experiment, TR2, we trained 5 SVM^{struct} models. The same features were used for both experiments. For the SVM algorithms, we generated 267,214 bag-of-word binary features. The Stanford Classifier and CRFsuite packages use internal routines to generate features.

The results of the experiments are summarized in Table 5. For TR1, our on-line ensemble methods improve over the best model. Note that $\mathcal{H}_{\text{SLE}}$ has the best average loss over 10 runs for this experiment. This comes at a price of much higher standard deviation which does not allow us to conclude that the difference in performance between our methods and $\mathcal{H}_{\text{SLE}}$ is statistically significant.

Table 6: Average Normalized Hamming Loss, SDS. $l \geq 4$, $\beta = 0.97$, $\delta = 0.05$, $T_{SLE} = 100$.

	$p = 5, m = 1500$	$p = 10, m = 1200$
$\mathcal{H}_{\text{MVote}}$	**0.2465 ± 0.00248**	**0.2606 ± 0.00320**
$\mathcal{H}_{\text{FPL}}$	0.2500 ± 0.00248	**0.2622 ± 0.00316**
$\mathcal{H}_{\text{CV}}$	0.2504 ± 0.00576	**0.2755 ± 0.00212**
$\mathcal{H}_{\text{FPL-CV}}$	0.2726 ± 0.00839	0.3219 ± 0.01176
$\mathcal{H}_{\text{ESPBoost}}$	0.2572 ± 0.00062	0.2864 ± 0.00103
$\mathcal{H}_{\text{SLE}}$	0.2572 ± 0.00061	0.2864 ± 0.00102
$\mathcal{H}_{\text{Rand}}$	0.2877 ± 0.00480	0.3430 ± 0.00468
Best h_j	0.2573 ± 0.00060	0.2865 ± 0.00101

Table 7: Average Normalized Hamming Loss, SDS. $l \geq 4$, $\beta = 0.97$,$\delta = 0.05$, $T_{SLE} = 100$.

	$p = 20, m = 900$	$p = 50, m = 700$
$\mathcal{H}_{\text{MVote}}$	**0.2773 ± 0.00139**	**0.3217 ± 0.00375**
$\mathcal{H}_{\text{FPL}}$	**0.2797 ± 0.00154**	**0.3189 ± 0.00344**
$\mathcal{H}_{\text{CV}}$	0.2986 ± 0.00075	0.3401 ± 0.00054
$\mathcal{H}_{\text{FPL-CV}}$	0.3816 ± 0.01457	0.4451 ± 0.01360
$\mathcal{H}_{\text{ESPBoost}}$	0.3115 ± 0.00089	0.3426 ± 0.00071
$\mathcal{H}_{\text{SLE}}$	0.3114 ± 0.00087	0.3425 ± 0.00076
$\mathcal{H}_{\text{Rand}}$	0.3977 ± 0.00302	0.4608 ± 0.00303
Best h_j	0.3116 ± 0.00087	0.3427 ± 0.00077

In fact, on two runs, $\mathcal{H}_{\text{SLE}}$ chooses an ensemble consisting of a single expert and fails to outperform the best model.

5.5 Speech recognition data set

For our last set of experiments, we used another proprietary speech recognition data set, SDS. Each example in this data set is represented by a sequence of length $l \in [2, 15]$. Therefore, for training we padded the true labels and the expert predictions to normalize the sequence lengths. For each of the 22,298 examples, there are between 2 and 251 expert predictions available. Since the ensemble methods we presented assume that the predictions of all p experts are available for each example in the training and test sets, we needed to restrict ourselves to the subsets of the data where at least some fixed number of expert predictions were available. In particular, we considered $p = 5, 10, 20$ and 50. For each value of p we used only the top p experts in our ensembles.

Our initial experiments showed that, as in the case of OCR data set, ensemble methods offer only a modest increase in performance over the best h_j. This is again largely due to the dominant performance of the best expert h_j. However, it was observed that the accuracy of the best model is a decreasing function of l, suggesting that ensemble algorithm may be used to improve performance for longer sequences. Subsequent experiments show that this is indeed the case: when training and testing with $l \geq 4$, ensemble algorithms outperform the best model. Table 6 and Table 7 summarize these results for $p = 5, 10, 20, 50$.

Our results suggest that the following simple scheme can be used: for short sequences use the best expert model and for longer sequences, use the ensemble model. A more elaborate variant of this algorithm can be derived based on the observation that the improvement in accuracy of the ensemble model over the best expert increases with the number of experts available.

6 Conclusion

We presented a broad analysis of the problem of ensemble structured prediction, including a series of algorithms with learning guarantees and extensive experiments. Our results show that our algorithms, most notably $\mathcal{H}_{\text{MVote}}$, can result in significant benefits in several tasks, which can be of a critical practical importance. We also reported very favorable results for $\mathcal{H}_{\text{MVote}}$ when used with the edit-distance, which is the standard loss used in many applications. A natural extension of this work consists of devising new algorithms and providing learning guarantees specific to other loss functions such as the edit-distance. While we aimed for an exhaustive study, including multiple on-learning algorithms, different conversions to batch and derandomizations, we are aware that the problem we studied is very rich and admits many more facets and scenarios that we plan to investigate in the future. Finally, the boosting-style algorithm we presented can be enhanced using recent theoretical and algorithmic results on *deep boosting* (Cortes et al., 2014b).

Acknowledgments

We warmly thank our colleagues Francoise Beaufays and Fuchun Peng for kindly extracting and making available to us the pronunciation data sets, Cyril Allauzen for providing us with the speech recognition data, and Richard Sproat and Brian Roark for help with other data sets. This work was partly funded by the NSF award IIS-1117591 and the NSERC PGS D3 award.

References

[Breiman1996] Leo Breiman. 1996. Bagging predictors. *Machine Learning*, 24(2):123–140.

[Caruana et al.2004] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes. 2004. Ensemble selection from libraries of models. In *Proceedings of ICML*, pages 18–.

[Cesa-Bianchi et al.2004] N. Cesa-Bianchi, A. Conconi, and C. Gentile. 2004. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057.

[Collins and Koo2005] Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70.

[Collins2002] M. Collins. 2002. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proceedings of ACL*, pages 1–8.

[Cortes et al.2005] C. Cortes, M. Mohri, and J. Weston. 2005. A general regression technique for learning transductions. In *Proceedings of ICML 2005*, pages 153–160, New York, NY, USA. ACM.

[Cortes et al.2014a] Corinna Cortes, Vitaly Kuznetsov, and Mehryar Mohri. 2014a. Ensemble methods for structured prediction. In *Proceedings of ICML*.

[Cortes et al.2014b] Corinna Cortes, Mehryar Mohri, and Umar Syed. 2014b. Deep boosting. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML 2014)*.

[Dekel and Singer2005] O. Dekel and Y. Singer. 2005. Data-driven online to batch conversion. In *Advances in NIPS 18*, pages 1207–1216.

[Fiscus1997] Jonathan G Fiscus. 1997. Post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover). In *Proceedings of the 1997 IEEE ASRU Workshop*, pages 347–354, Santa Barbara, CA.

[Freund and Schapire1997] Y. Freund and R. Schapire. 1997. A decision-theoretic generalization of on-line learning and application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139.

[Freund et al.2004] Yoav Freund, Yishay Mansour, and Robert E. Schapire. 2004. Generalization bounds for averaged classifiers. *The Annals of Statistics*, 32:1698–1722.

[Ghoshal et al.2009] Arnab Ghoshal, Martin Jansche, Sanjeev Khudanpur, Michael Riley, and Morgan Ulinski. 2009. Web-derived pronunciations. In *Proceedings of ICASSP*, pages 4289–4292.

[Joachims2008] T. Joachims. 2008. Support vector machines for complex outputs.

[Kalai and Vempala2005] A. Kalai and S. Vempala. 2005. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307.

[Kocev et al.2013] D. Kocev, C. Vens, J. Struyf, and S. Deroski. 2013. Tree ensembles for predicting structured outputs. *Pattern Recognition*, 46(3):817–833, March.

[Koltchinskii and Panchenko2002] Vladmir Koltchinskii and Dmitry Panchenko. 2002. Empirical margin distributions and bounding the generalization error of combined classifiers. *Annals of Statistics*, 30.

[Lafferty et al.2001] J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289.

[Littlestone and Warmuth1994] N. Littlestone and M. Warmuth. 1994. The weighted majority algorithm. *Information and Computation*, 108(2):212–261.

[Littlestone1989] N. Littlestone. 1989. From on-line to batch learning. In *Proceedings of COLT 2*, pages 269–284.

[MacKay1991] David J. C. MacKay. 1991. *Bayesian methods for adaptive models*. Ph.D. thesis, California Institute of Technology.

[MacKay1997] David J.C. MacKay. 1997. Ensemble learning for hidden markov models. Technical report, Cavendish Laboratory, Cambridge UK.

[Mohri et al.2008] Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. 2008. Speech recognition with weighted finite-state transducers. In *Handbook on Speech Processing and Speech Communication, Part E: Speech recognition*. Springer-Verlag.

[Mohri et al.2012] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. 2012. *Foundations of Machine Learning*. The MIT Press.

[Mohri1997] Mehryar Mohri. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311.

[Nguyen and Guo2007] N. Nguyen and Y. Guo. 2007. Comparison of sequence labeling algorithms and extensions. In *Proceedings of ICML*, pages 681–688.

[Okazaki2007] N. Okazaki. 2007. CRFsuite: a fast implementation of conditional random fields (crfs).

[Petrov2010] Slav Petrov. 2010. Products of random latent variable grammars. In *HLT-NAACL*, pages 19–27.

[Rafferty et al.2014] A. Rafferty, A. Kleeman, J. Finkel, and C. Manning. 2014. Stanford classifer.

[Sagae and Lavie2006] K. Sagae and A. Lavie. 2006. Parser combination by reparsing. In *Proceedings of HLT/NAACL*, pages 129–132.

[Schapire and Singer1999] Robert E. Schapire and Yoram Singer. 1999. Improved boosting algorithms

using confidence-rated predictions. *Machine Learning*, 37(3):297–336.

[Schapire and Singer2000] Robert E. Schapire and Yoram Singer. 2000. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2-3):135–168.

[Schapire et al.1997] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. 1997. Boosting the margin: A new explanation for the effectiveness of voting methods. In *ICML*, pages 322–330.

[Smyth and Wolpert1999] Padhraic Smyth and David Wolpert. 1999. Linearly combining density estimators via stacking. *Machine Learning*, 36:59–83, July.

[Takimoto and Warmuth2003] E. Takimoto and M. K. Warmuth. 2003. Path kernels and multiplicative updates. *JMLR*, 4:773–818.

[Taskar et al.2004] B. Taskar, C. Guestrin, and D. Koller. 2004. Max-margin Markov networks. In *Advances in NIPS 16*. MIT Press, Cambridge, MA.

[Tsochantaridis et al.2005] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. 2005. Large margin methods for structured and interdependent output variables. *JMLR*, 6:1453–1484, December.

[Wang et al.2007] Q. Wang, D. Lin, and D. Schuurmans. 2007. Simple training of dependency parsers via structured boosting. In *Proceedings of IJCAI 20*, pages 1756–1762.

[Zeman and Žabokrtský2005] D. Zeman and Z. Žabokrtský. 2005. Improving parsing accuracy by combining diverse dependency parsers. In *Proceedings of IWPT 9*, pages 171–178.

[Zhang et al.2009] H. Zhang, M. Zhang, C. Tan, and H. Li. 2009. K-best combination of syntactic parsers. In *Proceedings of EMNLP: Volume 3*, pages 1552–1560.

Representation Learning for Text-level Discourse Parsing

Yangfeng Ji
School of Interactive Computing
Georgia Institute of Technology
jiyfeng@gatech.edu

Jacob Eisenstein
School of Interactive Computing
Georgia Institute of Technology
jacobe@gatech.edu

Abstract

Text-level discourse parsing is notoriously difficult, as distinctions between discourse relations require subtle semantic judgments that are not easily captured using standard features. In this paper, we present a representation learning approach, in which we transform surface features into a latent space that facilitates RST discourse parsing. By combining the machinery of large-margin transition-based structured prediction with representation learning, our method jointly learns to parse discourse while at the same time learning a discourse-driven projection of surface features. The resulting shift-reduce discourse parser obtains substantial improvements over the previous state-of-the-art in predicting relations and nuclearity on the RST Treebank.

1 Introduction

Discourse structure describes the high-level organization of text or speech. It is central to a number of high-impact applications, such as text summarization (Louis et al., 2010), sentiment analysis (Voll and Taboada, 2007; Somasundaran et al., 2009), question answering (Ferrucci et al., 2010), and automatic evaluation of student writing (Miltsakaki and Kukich, 2004; Burstein et al., 2013). Hierarchical discourse representations such as Rhetorical Structure Theory (RST) are particularly useful because of the computational applicability of tree-shaped discourse structures (Taboada and Mann, 2006), as shown in Figure 1.

Unfortunately, the performance of discourse parsing is still relatively weak: the state-of-the-art F-measure for text-level relation detection in the RST Treebank is only slightly above 55% (Joty et al., 2013). While recent work has introduced increasingly powerful features (Feng and Hirst, 2012) and inference techniques (Joty et al., 2013), discourse relations remain hard to detect, due in part to a long tail of "alternative lexicalizations" that can be used to realize each relation (Prasad et al., 2010). Surface and syntactic features are not capable of capturing what are fundamentally semantic distinctions, particularly in the face of relatively small annotated training sets.

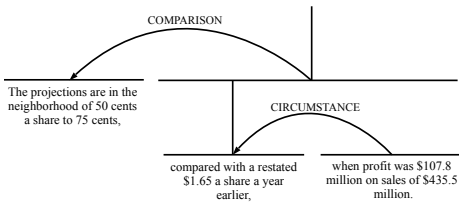


Figure 1: An example of RST discourse structure.

In this paper, we present a representation learning approach to discourse parsing. The core idea of our work is to learn a transformation from a bag-of-words surface representation into a latent space in which discourse relations are easily identifiable. The latent representation for each discourse unit can be viewed as a discriminatively-trained vector-space representation of its meaning. Alternatively, our approach can be seen as a non-linear learning algorithm for incremental structure prediction, which overcomes feature sparsity through effective parameter tying. We consider several alternative methods for transforming the original features, corresponding to different ideas of the meaning and role of the latent representation.

Our method is implemented as a shift-reduce discourse parser (Marcu, 1999; Sagae, 2009). Learning is performed as large-margin transition-based structure prediction (Taskar et al., 2003), while at the same time jointly learning to project the surface representation into latent space. The

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 13–24,
Baltimore, Maryland, USA, June 23-25 2014.

resulting system strongly outperforms the prior state-of-the-art at labeled F-measure, obtaining raw improvements of roughly 6% on relation labels and 2.5% on nuclearity. In addition, we show that the latent representation coheres well with the characterization of discourse connectives in the Penn Discourse Treebank (Prasad et al., 2008).

2 Model

The core idea of this paper is to project lexical features into a latent space that facilitates discourse parsing. In this way, we can capture the meaning of each discourse unit, without suffering from the very high dimensionality of a lexical representation. While such feature learning approaches have proven to increase robustness for parsing, POS tagging, and NER (Miller et al., 2004; Koo et al., 2008; Turian et al., 2010), they would seem to have an especially promising role for discourse, where training data is relatively sparse and ambiguity is considerable. Prasad et al. (2010) show that there is a long tail of alternative lexicalizations for discourse relations in the Penn Discourse Treebank, posing obvious challenges for approaches based on directly matching lexical features observed in the training data.

Based on this observation, our goal is to learn a function that transforms lexical features into a much lower-dimensional latent representation, while simultaneously learning to predict discourse structure based on this latent representation. In this paper, we consider a simple transformation function, linear projection. Thus, we name the approach DPLP: Discourse Parsing from Linear Projection. We apply transition-based (incremental) structured prediction to obtain a discourse parse, training a predictor to make the correct incremental moves to match the annotations of training data in the RST Treebank. This supervision signal is then used to learn both the weights and the projection matrix in a large-margin framework.

2.1 Shift-reduce discourse parsing

We construct RST Trees using shift-reduce parsing, as first proposed by Marcu (1999). At each point in the parsing process, we maintain a stack and a queue; initially the stack is empty and the first elementary discourse unit (EDU) in the document is at the front of the queue.[1] The parser can then choose either to *shift* the front of the queue onto the top of the stack, or to *reduce* the top two elements on the stack in a discourse relation. The reduction operation must choose both the type of relation and which element will be the nucleus. So, overall there are multiple reduce operations with specific relation types and nucleus positions. Shift-reduce parsing can be learned as a classification task, where the classifier uses features of the elements in the stack and queue to decide what move to take. Previous work has employed decision trees (Marcu, 1999) and the averaged perceptron (Collins and Roark, 2004; Sagae, 2009) for this purpose. Instead, we employ a large-margin classifier, because we can compute derivatives of the margin-based objective function with respect to both the classifier weights as well as the projection matrix.

Notation	Explanation
$\mathcal{V}$	Vocabulary for surface features
V	Size of $\mathcal{V}$
K	Dimension of latent space
$\mathbf{w}_m$	Classification weights for class m
C	Total number of classes, which correspond to possible shift-reduce operations
$\mathbf{A}$	Parameter of the representation function (also the projection matrix in the linear representation function)
$\mathbf{v}_i$	Word count vector of discourse unit i
$\mathbf{v}$	Vertical concatenation of word count vectors for the three discourse units currently being considered by the parser
λ	Regularization for classification weights
τ	Regularization for projection matrix
ξ_i	Slack variable for sample i
$\eta_{i,m}$	Dual variable for sample i and class m
α_t	Learning rate at iteration t

Table 1: Summary of mathematical notation

2.2 Discourse parsing with projected features

More formally, we denote the surface feature vocabulary $\mathcal{V}$, and represent each EDU as the numeric vector $\mathbf{v} \in \mathbb{N}^V$, where $V = \#|\mathcal{V}|$ and the n-th element of $\mathbf{v}$ is the count of the n-th surface feature in this EDU (see Table 1 for a summary of notation). During shift-reduce parsing, we consider features of three EDUs:[2] the top two elements on

[1] We do not address segmentation of text into elementary discourse units in this paper. Standard classification-based approaches can achieve a segmentation F-measure of 94% (Hernault et al., 2010); a more complex reranking model does slightly better, at 95% F-Measure with automatically-generated parse trees, and 96.6% with gold annotated trees (Xuan Bach et al., 2012). Human agreement reaches 98% F-Measure.

[2] After applying a reduce operation, the stack will include a span that contains multiple EDUs. We follow the *strong*

the stack ($\mathbf{v}_1$ and $\mathbf{v}_2$), and the front of the queue ($\mathbf{v}_3$). The vertical concatenation of these vectors is denoted $\mathbf{v} = [\mathbf{v}_1; \mathbf{v}_2; \mathbf{v}_3]$. In general, we can formulate the decision function for the multi-class shift-reduce classifier as

$$\hat{m} = \underset{m \in \{1,\ldots,C\}}{\arg\max}\ \mathbf{w}_m^\top \mathbf{f}(\mathbf{v}; \mathbf{A}) \quad (1)$$

where $\mathbf{w}_m$ is the weight for the m-th class and $\mathbf{f}(\mathbf{v}; \mathbf{A})$ is the *representation function* parametrized by $\mathbf{A}$. The score for class m (in our case, the value of taking the m-th shift-reduce operation) is computed by the inner product $\mathbf{w}_m^\top \mathbf{f}(\mathbf{v}; \mathbf{A})$. The specific shift-reduce operation is chosen by maximizing the decision value in Equation 1.

The representation function $\mathbf{f}(\mathbf{v}; \mathbf{A})$ can be defined in any form; for example, it could be a non-linear function defined by a neural network model parametrized by $\mathbf{A}$. We focus on the linear projection,

$$\mathbf{f}(\mathbf{v}; \mathbf{A}) = \mathbf{A}\mathbf{v}, \quad (2)$$

where $\mathbf{A} \in \mathbb{R}^{K \times 3V}$ is projects the surface representation $\mathbf{v}$ of three EDUs into a latent space of size $K \ll V$.

Note that by setting $\tilde{\mathbf{w}}_m^\top = \mathbf{w}_m^\top \mathbf{A}$, the decision scoring function can be rewritten as $\tilde{\mathbf{w}}_m^\top \mathbf{v}$, which is linear in the original surface features. Therefore, the expressiveness of DPLP is identical to a linear separator in the original feature space. However, the learning problem is considerably different. If there are C total classes (possible shift-reduce operations), then a linear classifier must learn $3VC$ parameters, while DPLP must learn $(3V + C)K$ parameters, which will be smaller under the assumption that $K < C \ll V$. This can be seen as a form of *parameter tying* on the linear weights $\tilde{\mathbf{w}}_m$, which allows statistical strength to be shared across training instances. We will consider special cases of $\mathbf{A}$ that reduce the parameter space still further.

2.3 Special forms of the projection matrix

We consider three different constructions for the projection matrix $\mathbf{A}$.

- *General form*: In the general case, we place no special constraint on the form of $\mathbf{A}$.

$$\mathbf{f}(\mathbf{v}; \mathbf{A}) = \mathbf{A} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \end{bmatrix} \quad (3)$$

  This form is shown in Figure 2(a).

- *Concatenation form*: In the concatenation form, we choose a block structure for $\mathbf{A}$, in which a single projection matrix $\mathbf{B}$ is applied to each EDU:

$$\mathbf{f}(\mathbf{v}; \mathbf{A}) = \begin{bmatrix} \mathbf{B} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{B} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{B} \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \end{bmatrix} \quad (4)$$

  In this form, we transform the representation of each EDU separately, but do not attempt to represent interrelationships between the EDUs in the latent space. The number of parameters in $\mathbf{A}$ is $\frac{1}{3}KV$. Then, the total number of parameters, including the decision weights $\{\mathbf{w}_m\}$, in this form is $(\frac{V}{3} + C)K$.

- *Difference form*. In the difference form, we explicitly represent the *differences* between adjacent EDUs, by constructing $\mathbf{A}$ as a block difference matrix,

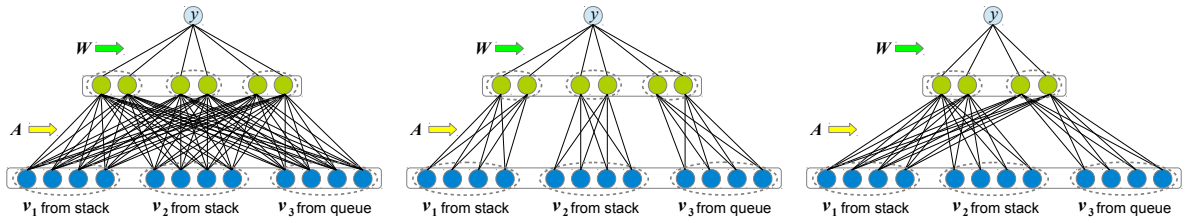
$$\mathbf{f}(\mathbf{v}; \mathbf{A}) = \begin{bmatrix} \mathbf{C} & -\mathbf{C} & \mathbf{0} \\ \mathbf{C} & \mathbf{0} & -\mathbf{C} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \end{bmatrix}, \quad (5)$$

  The result of this projection is that the latent representation has the form $[\mathbf{C}(\mathbf{v}_1 - \mathbf{v}_2); \mathbf{C}(\mathbf{v}_1 - \mathbf{v}_3)]$, representing the difference between the top two EDUs on the stack, and between the top EDU on the stack and the first EDU in the queue. This is intended to capture semantic similarity, so that reductions between related EDUs will be preferred. Similarly, the total number of parameters to estimate in this form is $(V + 2C)\frac{K}{3}$.

3 Large-Margin Learning Framework

We apply a large margin structure prediction approach to train the model. There are two parameters that need to be learned: the classification weights $\{\mathbf{w}_m\}$, and the projection matrix $\mathbf{A}$. As we will see, it is possible to learn $\{\mathbf{w}_m\}$ using standard support vector machine (SVM) training (holding $\mathbf{A}$ fixed), and then make a simple gradient-based update to $\mathbf{A}$ (holding $\{\mathbf{w}_m\}$ fixed). By interleaving these two operations, we arrive at a saddle point of the objective function.

compositionality criterion of Marcu (1996) and consider only the nuclear EDU of the span. Later work may explore the composition of features between the nucleus and satellite.



(a) General form (b) Concatenation form (c) Difference form

Figure 2: Decision problem with different representation functions

Specifically, we formulate the following constrained optimization problem,

$$\begin{aligned} \min_{\{\mathbf{w}_{1:C},\xi_{1:l},\mathbf{A}\}} & \frac{\lambda}{2}\sum_{m=1}^{C}\|\mathbf{w}_m\|_2^2+\sum_{i=1}^{l}\xi_i+\frac{\tau}{2}\|\mathbf{A}\|_F^2 \\ \text{s.t. } & (\mathbf{w}_{y_i}-\mathbf{w}_m)^\top\mathbf{f}(\mathbf{v}_i;\mathbf{A})\geq 1-\delta_{y_i=m}-\xi_i, \\ & \forall\, i,m \end{aligned} \tag{6}$$

where $m \in \{1,\ldots,C\}$ is the index of the shift-reduce decision taken by the classifier (e.g., SHIFT, REDUCE-CONTRAST-RIGHT, etc), $i \in \{1,\cdots,l\}$ is the index of the training sample, and $\mathbf{w}_m$ is the vector of classification weights for class m. The slack variables ξ_i permit the margin constraint to be violated in exchange for a penalty, and the delta function $\delta_{y_i=m}$ is unity if $y_i = m$, and zero otherwise.

As is standard in the multi-class linear SVM (Crammer and Singer, 2001), we can solve the problem defined in Equation 6 via Lagrangian optimization:

$$\begin{aligned} &\mathcal{L}(\{\mathbf{w}_{1:C},\xi_{1:l},\mathbf{A},\eta_{1:l,1:C}\}) = \\ &\quad\frac{\lambda}{2}\sum_{m=1}^{C}\|\mathbf{w}_m\|_2^2+\sum_{i=1}^{l}\xi_i+\frac{\tau}{2}\|\mathbf{A}\|_F^2 \\ &\quad+\sum_{i,m}\eta_{i,m}\Big\{(\mathbf{w}_m^\top-\mathbf{w}_{y_i}^\top)\mathbf{f}(\mathbf{v}_i;\mathbf{A})+1-\delta_{y_i=m}-\xi_i\Big\} \\ &\quad\text{s.t. } \eta_{i,m}\geq 0\ \forall i,m \end{aligned} \tag{7}$$

Then, to optimize $\mathcal{L}$, we need to find a saddle point, which would be the minimum for the variables $\{\mathbf{w}_{1:C},\xi_{1:l}\}$ and the projection matrix $\mathbf{A}$, and the maximum for the dual variables $\{\eta_{1:l,1:C}\}$.

If $\mathbf{A}$ is fixed, then the optimization problem is equivalent to a standard multi-class SVM, in the transformed feature space $\mathbf{f}(\mathbf{v}_i;\mathbf{A})$. We can obtain the weights $\{\mathbf{w}_{1:C}\}$ and dual variables $\{\eta_{1:l,1:C}\}$ from a standard dual-form SVM solver. We then update $\mathbf{A}$, recompute $\{\mathbf{w}_{1:C}\}$ and $\{\eta_{1:l,1:C}\}$, and iterate until convergence. This iterative procedure is similar to the latent variable structural SVM (Yu and Joachims, 2009), although the specific details of our learning algorithm are different.

3.1 Learning Projection Matrix A

We update $\mathbf{A}$ while holding fixed the weights and dual variables. The derivative of $\mathcal{L}$ with respect to $\mathbf{A}$ is

$$\begin{aligned} \frac{\partial\mathcal{L}}{\partial\mathbf{A}} &= \tau\mathbf{A}+\sum_{i,m}\eta_{i,m}(\mathbf{w}_m^\top-\mathbf{w}_{y_i}^\top)\frac{\partial\mathbf{f}(\mathbf{v}_i;\mathbf{A})}{\partial\mathbf{A}} \\ &= \tau\mathbf{A}+\sum_{i,m}\eta_{i,m}(\mathbf{w}_m-\mathbf{w}_{y_i})\mathbf{v}_i{}^\top \end{aligned} \tag{8}$$

Setting $\frac{\partial\mathcal{L}}{\partial\mathbf{A}}=0$, we have the closed-form solution,

$$\begin{aligned} \mathbf{A} &= \frac{1}{\tau}\sum_{i,m}\eta_{i,m}(\mathbf{w}_m-\mathbf{w}_{y_i})\mathbf{v}_i{}^\top \\ &= \frac{1}{\tau}\sum_{i,j}(\mathbf{w}_{y_i}-\sum_m\eta_{i,m}\mathbf{w}_m)\mathbf{v}_i{}^\top, \end{aligned} \tag{9}$$

because the dual variables for each instance must sum to one, $\sum_m\eta_{i,m}=1$.

Note that for a given i, the matrix $(\mathbf{w}_{y_i}-\sum_m\eta_{i,m}\mathbf{w}_m)\mathbf{v}_i{}^\top$ is of (at most) rank-1. Therefore, the solution of $\mathbf{A}$ can be viewed as the linear combination of a sequence of rank-1 matrices, where each rank-1 matrix is defined by distributional representation $\mathbf{v}_i$ and the weight difference between the weight of true label $\mathbf{w}_{y_i}$ and the "expected" weight $\sum_m\eta_{i,m}\mathbf{w}_m$.

One property of the dual variables is that $\mathbf{f}(\mathbf{v}_i;\mathbf{A})$ is a support vector only if the dual variable $\eta_{i,y_i}<1$. Since the dual variables for each instance are guaranteed to sum to one, we have $\mathbf{w}_{y_i}-\sum_m\eta_{i,m}\mathbf{w}_m=0$ if $\eta_{i,y_i}=1$. In other words, the contribution from non support vectors to the projection matrix $\mathbf{A}$ is 0. Then, we can further simplify the updating equation as

$$\mathbf{A}=\frac{1}{\tau}\sum_{\mathbf{v}_i\in\mathrm{SV}}(\mathbf{w}_{y_i}-\sum_m\eta_{i,m}\mathbf{w}_m)\mathbf{v}_i{}^\top \tag{10}$$

This is computationally advantageous since many instances are not support vectors, and it shows that the discriminatively-trained projection matrix only incorporates information from each instance to the extent that the correct classification receives low confidence.

Algorithm 1 Mini-batch learning algorithm

Input: Training set $\mathcal{D}$, Regularization parameters λ and τ, Number of iteration T, Initialization matrix $\mathbf{A}_0$, and Threshold ε
while $t = 1, \ldots, T$ **do**
  Randomly choose a subset of training samples $\mathcal{D}_t$ from $\mathcal{D}$
  Train SVM with $\mathbf{A}_{t-1}$ to obtain $\{\mathbf{w}_m^{(t)}\}$ and $\{\eta_{i,m}^{(t)}\}$
  Update $\mathbf{A}_t$ using Equation 11 with $\alpha_t = \frac{1}{t}$
  if $\frac{\|\mathbf{A}_t - \mathbf{A}_{t-1}\|_F}{\|\mathbf{A}_2 - \mathbf{A}_1\|_F} < \varepsilon$ **then**
    Return
  end if
end while
Re-train SVM with $\mathcal{D}$ and the final $\mathbf{A}$
Output: Projection matrix $\mathbf{A}$, SVM classifier with weights $\mathbf{w}$

3.2 Gradient-based Learning for A

Solving the quadratic programming defined by the dual form of the SVM is time-consuming, especially on a large-scale dataset. But if we focus on learning the projection matrix $\mathbf{A}$, we can speed up learning by sampling only a small proportion of the training data to compute an approximate optimum for $\{\mathbf{w}_{1:C}, \eta_{1:l,1:C}\}$, before each update of $\mathbf{A}$. This idea is similar to the mini-batch learning, which has been used in large-scale SVM problem (Nelakanti et al., 2013) and deep learning models (Le et al., 2011).

Specifically, in iteration t, the algorithm randomly chooses a subset of training samples $\mathcal{D}_t$ to train the model. We cannot make a closed-form update to $\mathbf{A}$ based on this small sample, but we can take an approximate gradient step,

$$\mathbf{A}_t = (1 - \alpha_t \tau)\mathbf{A}_{t-1} + \alpha_t \Big\{ \sum_{\mathbf{v}_i \in \mathrm{SV}(\mathcal{D}_t)} \Big(\mathbf{w}_{y_i}^{(t)} - \sum_m \eta_{i,m}^{(t)} \mathbf{w}_m^{(t)} \Big) {\mathbf{v}_i}^\top \Big\}, \quad (11)$$

where α_t is a learning rate. In iteration t, we choose $\alpha_t = \frac{1}{t}$. After convergence, we obtain the weights $\mathbf{w}$ by applying the SVM over the entire dataset, using the final $\mathbf{A}$. The algorithm is summarized in Algorithm 1 and more details about implementation will be clarified in Section 4. While minibatch learning requires more iterations, the SVM training is much faster in each batch, and the overall algorithm is several times faster than using the entire training set for each update.

4 Implementation

The learning algorithm is applied in a shift-reduce parser, where the training data consists of the (unique) list of shift and reduce operations required to produce the gold RST parses. On test data, we choose parsing operations in an online fashion — at each step, the parsing algorithm changes the status of the stack and the queue according the selected transition, then creates the next sample with the updated status.

4.1 Parameters and Initialization

There are three free parameters in our approach: the latent dimension K, and regularization parameters λ and τ. We consider the values $K \in \{30, 60, 90, 150\}$, $\lambda \in \{1, 10, 50, 100\}$ and $\tau \in \{1.0, 0.1, 0.01, 0.001\}$, and search over this space using a development set of thirty document randomly selected from within the RST Treebank training data. We initialize each element of $\mathbf{A}_0$ to a uniform random value in the range $[0, 1]$. For mini-batch learning, we fixed the batch size to be 500 training samples (shift-reduce operations) in each iteration.

4.2 Additional features

As described thus far, our model considers only the projected representation of each EDU in its parsing decisions. But prior work has shown that other, structural features can provide useful information (Joty et al., 2013). We therefore augment our classifier with a set of simple feature templates. These templates are applied to individual EDUs, as well as pairs of EDUs: (1) the two EDUs on top of the stack, and (2) the EDU on top of the stack and the EDU in front of the queue. The features are shown in Table 2. In computing these features, all tokens are downcased, and numerical features are not binned. The dependency structure and POS tags are obtained from MALTParser (Nivre et al., 2007).

5 Experiments

We evaluate DPLP on the RST Discourse Treebank (Carlson et al., 2001), comparing against state-of-the-art results. We also investigate the information encoded by the projection matrix.

5.1 Experimental Setup

Dataset The RST Discourse Treebank (RST-DT) consists of 385 documents, with 347 for train-

Feature	Examples
Words at beginning and end of the EDU	⟨BEGIN-WORD-STACK1 = *but*⟩ ⟨BEGIN-WORD-STACK1-QUEUE1 = *but, the*⟩
POS tag at beginning and end of the EDU	⟨BEGIN-TAG-STACK1 = CC⟩ ⟨BEGIN-TAG-STACK1-QUEUE1 = CC, DT⟩
Head word set from each EDU. The set includes words whose parent in the depenency graph is ROOT or is not within the EDU (Sagae, 2009).	⟨HEAD-WORDS-STACK2 = *working*⟩
Length of EDU in tokens	⟨LEN-STACK1-STACK2 = ⟨7, 8⟩⟩
Distance between EDUs	⟨DIST-STACK1-QUEUE1 = 2⟩
Distance from the EDU to the beginning of the document	⟨DIST-FROM-START-QUEUE1 = 3⟩
Distance from the EDU to the end of the document	⟨DIST-FROM-END-STACK1 = 1⟩
Whether two EDUs are in the same sentence	⟨SAME-SENT-STACK1-QUEUE1 = True⟩

Table 2: Additional features for RST parsing

ing and 38 for testing in the standard split. As we focus on relational discourse parsing, we follow prior work (Feng and Hirst, 2012; Joty et al., 2013), and use gold EDU segmentations. The strongest automated RST segmentation methods currently attain 95% accuracy (Xuan Bach et al., 2012).

Preprocessing In the RST-DT, most nodes have exactly two children, one nucleus and one satellite. For non-binary relations, we use right-branching to binarize the tree structure. For multi-nuclear relations, we choose the left EDU as "head" EDU. The vocabulary $\mathcal{V}$ includes all unigrams after down-casing. No other preprocessing is performed. In total, there are 16250 unique unigrams in $\mathcal{V}$.

Fixed projection matrix baselines Instead of learning from data, a simple way to obtain a projection matrix is to use matrix factorization. Recent work has demonstrated the effectiveness of non-negative matrix factorization (NMF) for measuring distributional similarity (Dinu and Lapata, 2010; Van de Cruys and Apidianaki, 2011). We can construct $\mathbf{B}_{nmf}$ in the *concatenation form* of the projection matrix by applying NMF to the EDU-feature matrix, $\mathbf{M} \approx \mathbf{WH}$. As a result, $\mathbf{W}$ describes each EDU with a K-dimensional vector, and $\mathbf{H}$ describes each word with a K-dimensional vector. We can then construct $\mathbf{B}_{nmf}$ by taking the pseudo-inverse of $\mathbf{H}$, which then projects from word-count vectors into the latent space.

Another way to construct $\mathbf{B}$ is to use neural word embeddings (Collobert and Weston, 2008). In this case, we can view the product $\mathbf{Bv}$ as a composition of the word embeddings, using the simple *additive* composition model proposed by Mitchell and Lapata (2010). We used the word embeddings from Collobert and Weston (2008) with dimension $\{25, 50, 100\}$. Grid search over heldout training data was used to select the optimum latent dimension for both the NMF and word embedding baselines. Note that the size K of the resulting projection matrix is three times the size of the embedding (or NMF representation) due to the concatenate construction.

We also consider the special case where $\mathbf{A} = \mathbf{I}$.

Competitive systems We compare our approach with HILDA (Hernault et al., 2010) and TSP (Joty et al., 2013). Joty et al. (2013) proposed two different approaches to combine sentence-level parsing models: *sliding windows* (TSP SW) and *1 sentence-1 subtree* (TSP 1-1). In the comparison, we report the results of both approaches. All results are based on the same gold standard EDU segmentation. We cannot compare with the results of Feng and Hirst (2012), because they do not evaluate on the overall discourse *structure*, but rather treat each relation as an individual classification problem.

Metrics To evaluate the parsing performance, we use the three standard ways to measure the performance: unlabeled (i.e., hierarchical spans) and labeled (i.e., nuclearity and relation) F-score, as defined by Black et al. (1991). The application of this approach to RST parsing is described by Marcu (2000b).[3] To compare with previous works on RST-DT, we use the 18 coarse-grained relations defined in (Carlson et al., 2001).

[3]We implemented the evaluation metrics by ourselves. Together with the DPLP system, all codes are published on `https://github.com/jiyfeng/DPLP`

Method	Matrix Form	+Features	K	Span	Nuclearity	Relation
Prior work						
1. HILDA (Hernault *et al.*, 2010)				**83.0**	68.4	54.8
2. TSP 1-1 (Joty *et al.*, 2013)				82.47	68.43	55.73
3. TSP SW (Joty *et al.*, 2013)				82.74	68.40	55.71
Our work						
4. Basic features	$\mathbf{A} = \mathbf{0}$	Yes		79.43	67.98	52.96
5. Word embeddings	Concatenation	No	75	75.28	67.14	53.79
6. NMF	Concatenation	No	150	78.57	67.66	54.80
7. Bag-of-words	$\mathbf{A} = \mathbf{I}$	Yes		79.85	69.01	60.21
8. DPLP	Concatenation	No	60	80.91	69.39	58.96
9. DPLP	Difference	No	60	80.47	68.61	58.27
10. DPLP	Concatenation	Yes	60	82.08	**71.13**	**61.63**
11. DPLP	General	Yes	30	81.60	**70.95**	**61.75**
Human annotation				88.70	77.72	65.75

Table 3: Parsing results of different models on the RST-DT test set. The results of TSP and HILDA are reprinted from prior work (Joty et al., 2013; Hernault et al., 2010).

5.2 Experimental Results

Table 3 presents RST parsing results for DPLP and some alternative systems. All versions of DPLP outperform the prior state-of-the-art on nuclearity and relation detection. This includes relatively simple systems whose features are simply a projection of the word count vectors for each EDU (lines 7 and 8). The addition of the features from Table 2 improves performance further, leading to absolute F-score improvement of around 2.5% in nuclearity and 6% in relation prediction (lines 9 and 10).

On span detection, DPLP performs slightly worse than the prior state-of-the-art. These systems employ richer syntactic and contextual features, which might be especially helpful for span identification. As shown by line 4 of the results table, the basic features from Table 2 provide most of the predictive power for spans; however, these features are inadequate at the more semantically-oriented tasks of nuclearity and relation prediction, which benefit substantially from the projected features. Since correctly identifying spans is a precondition for nuclearity and relation prediction, we might obtain still better results by combining features from HILDA and TSP with the representation learning approach described here.

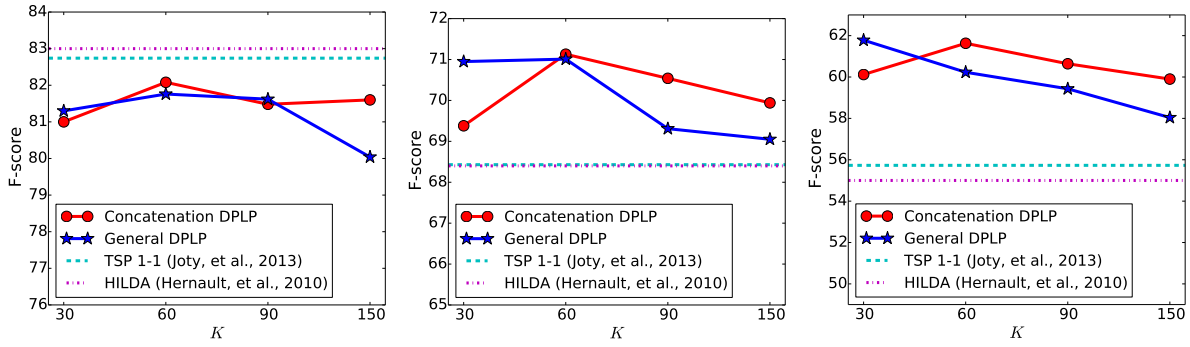
Lines 5 and 6 show that discriminative learning of the projection matrix is crucial, as fixed projections obtained from NMF or neural word embeddings perform substantially worse. Line 7 shows that the original bag-of-words representation together with basic features could give us some benefit on discourse parsing, but still not as good as results from DPLP. From lines 8 and 9, we see that the concatenation construction is superior to the difference construction, but the comparison between lines 10 and 11 is inconclusive on the merits of the general form of $\mathbf{A}$. This suggests that using the projection matrix to model interrelationships between EDUs does not substantially improve performance, and the simpler concatenation construction may be preferred.

Figure 3 shows how performance changes for different latent dimensions K. At each value of K, we employ grid search over a development set to identify the optimal regularizers λ and τ. For the concatenation construction, performance is not overly sensitive to K. For the general form of $\mathbf{A}$, performance decreases with large K. Recall from Section 2.3 that this construction has nine times as many parameters as the concatenation form; with large values of K, it is likely to overfit.

5.3 Analysis of Projection Matrix

Why does projection of the surface features improve discourse parsing? To answer this question, we examine what information the projection matrix is learning to encoded. We take the projection matrix from the concatenation construction and $K = 60$ as an example for case study. Recalling the definition in equation 4, the projection matrix $\mathbf{A}$ will be composed of three identical submatrices $\mathbf{B} \in \mathbb{R}^{20 \times V}$. The columns of the $\mathbf{B}$ matrix can be viewed as 20-dimensional descriptors of the words in the vocabulary.

For the purpose of visualization, we further reduce the dimension of latent representation from $K = 20$ to 2 dimensions using t-SNE (van der Maaten and Hinton, 2008). One further simpli-



(a) Span (b) Nuclearity (c) Relation

Figure 3: The performance of our parser over different latent dimension K. Results for DPLP include the additional features from Table 3

fication for visualization is we consider only the top 1000 frequent unigrams in the RST-DT training set. For comparison, we also apply t-SNE to the projection matrix $\mathbf{B}_{nmf}$ recovered from non-negative matrix factorization.

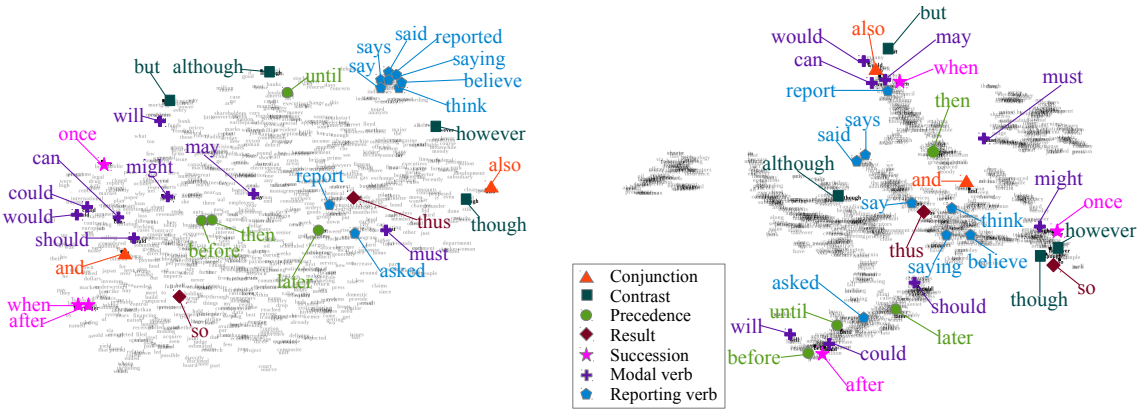
Figure 4 highlights words that are related to discourse analysis. Among the top 1000 words, we highlight the words from 5 major discourse connective categories provided in Appendix B of the PDTB annotation manual (Prasad et al., 2008): CONJUNCTION, CONTRAST, PRECEDENCE, RESULT, and SUCCESSION. In addition, we also highlighted two verb categories from the top 1000 words: modal verbs and reporting verbs, with their inflections (Krestel et al., 2008).

From the figure, it is clear DPLP has learned a projection matrix that successfully groups several major discourse-related word classes: particularly modal and reporting verbs; it has also grouped succession and precedence connectives with some success. In contrast, while NMF does obtain compact clusters of words, these clusters appear to be completely unrelated to discourse function of the words that they include. This demonstrates the value of using discriminative training to obtain the transformed representation of the discourse units.

6 Related Work

Early work on document-level discourse parsing applied hand-crafted rules and heuristics to build trees in the framework of Rhetorical Structure Theory (Sumita et al., 1992; Corston-Oliver, 1998; Marcu, 2000a). An early data-driven approach was offered by Schilder (2002), who used distributional techniques to rate the topicality of each discourse unit, and then chose among underspecified discourse structures by placing more topical sentences near the root. Learning-based approaches were first applied to identify within-sentence discourse relations (Soricut and Marcu, 2003), and only later to cross-sentence relations at the document level (Baldridge and Lascarides, 2005). Of particular relevance to our inference technique are incremental discourse parsing approaches, such as shift-reduce (Sagae, 2009) and A* (Muller et al., 2012). Prior learning-based work has largely focused on lexical, syntactic, and structural features, but the close relationship between discourse structure and semantics (Forbes-Riley et al., 2006) suggests that shallow feature sets may struggle to capture the long tail of alternative lexicalizations that can be used to realize discourse relations (Prasad et al., 2010; Marcu and Echihabi, 2002). Only Subba and Di Eugenio (2009) incorporate rich compositional semantics into discourse parsing, but due to the ambiguity of their semantic parser, they must manually select the correct semantic parse from a forest of possiblities.

Recent work has succeeded in pushing the state-of-the-art in RST parsing by innovating on several fronts. Feng and Hirst (2012) explore rich linguistic linguistic features, including lexical semantics and discourse production rules suggested by Lin et al. (2009) in the context of the Penn Discourse Treebank (Prasad et al., 2008). Muller et al. (2012) show that A* decoding can outperform both greedy and graph-based decoding algorithms. Joty et al. (2013) achieve the best prior results on RST relation detection by (i) jointly performing relation detection and classification, (ii) performing bottom-up rather than greedy decoding, and (iii) distinguishing between intra-sentence and inter-sentence relations. Our approach is largely orthogonal to this prior work: we focus on trans-



(a) Latent representation of words from projection learning with $K = 20$.

(b) Latent representation of words from non-negative matrix factorization with $K = 20$.

Figure 4: t-SNE Visualization on latent representations of words.

forming the lexical representation of discourse units into a latent space to facilitate learning. As shown in Figure 4(a), this projection succeeds at grouping words with similar discourse functions. We might expect to obtain further improvements by augmenting this representation learning approach with rich syntactic features (particularly for span identification), more accurate decoding, and special treatment of intra-sentence relations; this is a direction for future research.

Discriminative learning of latent features for discourse processing can be viewed as a form of *representation learning* (Bengio et al., 2013). Also called Deep Learning, such approaches have recently been applied in a number of NLP tasks (Collobert et al., 2011; Socher et al., 2012). Of particular relevance are applications to the detection of semantic or discourse relations, such as paraphrase, by comparing sentences in an induced latent space (Socher et al., 2011; Guo and Diab, 2012; Ji and Eisenstein, 2013). In this work, we show how discourse structure annotations can function as a supervision signal to discriminatively learn a transformation from lexical features to a latent space that is well-suited for discourse parsing. Unlike much of the prior work on representation learning, we induce a simple linear transformation. Extension of our approach by incorporating a non-linear activation function is a natural topic for future research.

7 Conclusion

We have presented a framework to perform discourse parsing while jointly learning to project to a low-dimensional representation of the discourse units. Using the vector-space representation of EDUs, our shift-reduce parsing system substantially outperforms existing systems on nuclearity detection and discourse relation identification. By adding some additional surface features, we obtain further improvements. The low dimensional representation also captures basic intuitions about discourse connectives and verbs, as shown in Figure 4(a).

Deep learning approaches typically apply a non-linear transformation such as the sigmoid function (Bengio et al., 2013). We have conducted a few unsuccessful experiments with the "hard tanh" function proposed by Collobert and Weston (2008), but a more complete exploration of non-linear transformations must wait for future work. Another direction would be more sophisticated composition of the surface features within each elementary discourse unit, such as the hierarchical convolutional neural network (Kalchbrenner and Blunsom, 2013) or the recursive tensor network (Socher et al., 2013). It seems likely that a better accounting for syntax could improve the latent representations that our method induces.

Acknowledgments

We thank the reviewers for their helpful feedback, particularly for the connection to multitask learning. We also want to thank Kenji Sagae and Vanessa Wei Feng for the helpful discussion via email communication. This research was supported by Google Faculty Research Awards to the second author.

References

Jason Baldridge and Alex Lascarides. 2005. Probabilistic head-driven parsing for discourse structure. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 96–103.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

Ezra Black, Steve Abney, Dan Flickinger, Claudia Gdaniec, Ralph Grishman, Phil Harrison, Don Hindle, Robert Ingria, Fred Jelinek, Judith Klavans, Mark Liberman, Mitchell Marcus, Salim Roukos, Beatrice Santorini, and Tomek Strzalkowski. 1991. A Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars. In *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19-22, 1991*, pages 306–311.

Jill Burstein, Joel Tetreault, and Martin Chodorow. 2013. Holistic discourse coherence annotation for noisy essay writing. *Dialogue & Discourse*, 4(2):34–52.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2001. Building a Discourse-tagged Corpus in the Framework of Rhetorical Structure Theory. In *Proceedings of Second SIGdial Workshop on Discourse and Dialogue*.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of ACL*, page 111. Association for Computational Linguistics.

R. Collobert and J. Weston. 2008. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *ICML*.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Simon Corston-Oliver. 1998. Beyond string matching and cue phrases: Improving efficiency and coverage in discourse analysis. In *The AAAI Spring Symposium on Intelligent Text Summarization*, pages 9–15.

Koby Crammer and Yoram Singer. 2001. On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines. *Journal of Machine Learning Research*, 2:265–292.

Georgiana Dinu and Mirella Lapata. 2010. Measuring Distributional Similarity in Context. In *EMNLP*, pages 1162–1172.

Vanessa Wei Feng and Graeme Hirst. 2012. Text-level Discourse Parsing with Rich Linguistic Features. In *Proceedings of ACL*.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. 2010. Building Watson: An overview of the DeepQA project. *AI magazine*, 31(3):59–79.

Katherine Forbes-Riley, Bonnie Webber, and Aravind Joshi. 2006. Computing discourse semantics: The predicate-argument semantics of discourse connectives in D-LTAG. *Journal of Semantics*, 23(1):55–106.

Weiwei Guo and Mona Diab. 2012. Modeling Sentences in the Latent Space. In *Proceedings of ACL*, pages 864–872, Jeju Island, Korea, July. Association for Computational Linguistics.

Hugo Hernault, Helmut Prendinger, David A. duVerle, and Mitsuru Ishizuka. 2010. HILDA: A Discourse Parser Using Support Vector Machine Classification. *Dialogue and Discourse*, 1(3):1–33.

Yangfeng Ji and Jacob Eisenstein. 2013. Discriminative Improvements to Distributional Sentence Similarity. In *EMNLP*, pages 891–896, Seattle, Washington, USA, October. Association for Computational Linguistics.

Shafiq Joty, Giuseppe Carenini, Raymond Ng, and Yashar Mehdad. 2013. Combining Intra- and Multi-sentential Rhetorical Parsing for Document-level Discourse Analysis. In *Proceedings of ACL*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, pages 119–126, Sofia, Bulgaria, August. Association for Computational Linguistics.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple Semi-supervised Dependency Parsing. In *Proceedings of ACL-HLT*, pages 595–603, Columbus, Ohio, June. Association for Computational Linguistics.

Ralf Krestel, Sabine Bergler, and René Witte. 2008. Minding the Source: Automatic Tagging of Reported Speech in Newspaper Articles. In *LREC*, Marrakech, Morocco, May. European Language Resources Association (ELRA).

Quoc V. Le, Jiquan Ngiam, Adam Coates, Abhik Lahiri, Bobby Prochnow, and Andrew Y. Ng. 2011. On Optimization Methods for Deep Learning. In *ICML*.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing Implicit Discourse Relations in the Penn Discourse Treebank. In *EMNLP*.

Annie Louis, Aravind Joshi, and Ani Nenkova. 2010. Discourse indicators for content selection in summarization. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 147–156. Association for Computational Linguistics.

Daniel Marcu and Abdessamad Echihabi. 2002. An Unsupervised Approach to Recognizing Discourse Relations. In *Proceedings of ACL*, pages 368–375, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Daniel Marcu. 1996. Building Up Rhetorical Structure Trees. In *Proceedings of AAAI*.

Daniel Marcu. 1999. A Decision-Based Approach to Rhetorical Parsing. In *Proceedings of ACL*, pages 365–372, College Park, Maryland, USA, June. Association for Computational Linguistics.

Daniel Marcu. 2000a. The Rhetorical Parsing of Unrestricted Texts: A Surface-based Approach. *Computational Linguistics*, 26:395–448.

Daniel Marcu. 2000b. *The Theory and Practice of Discourse Parsing and Summarization*. MIT Press.

Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name Tagging with Word Clusters and Discriminative Training. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL*, pages 337–342, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Eleni Miltsakaki and Karen Kukich. 2004. Evaluation of text coherence for electronic essay scoring systems. *Natural Language Engineering*, 10(1):25–55.

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

Philippe Muller, Stergos Afantenos, Pascal Denis, and Nicholas Asher. 2012. Constrained Decoding for Text-Level Discourse Parsing. In *Coling*, pages 1883–1900, Mumbai, India, December. The COLING 2012 Organizing Committee.

Anil Kumar Nelakanti, Cedric Archambeau, Julien Mairal, Francis Bach, and Guillaume Bouchard. 2013. Structured Penalties for Log-Linear Language Models. In *EMNLP*, pages 233–243, Seattle, Washington, USA, October. Association for Computational Linguistics.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The penn discourse treebank 2.0. In *LREC*.

Rashmi Prasad, Aravind Joshi, and Bonnie Webber. 2010. Realization of discourse relations by other means: alternative lexicalizations. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1023–1031. Association for Computational Linguistics.

Kenji Sagae. 2009. Analysis of Discourse Structure with Syntactic Dependencies and Data-Driven Shift-Reduce Parsing. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT)*, pages 81–84, Paris, France, October. Association for Computational Linguistics.

Frank Schilder. 2002. Robust discourse parsing via discourse markers, topicality and position. *Natural Language Engineering*, 8(3):235–255.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In *NIPS*.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic Compositionality Through Recursive Matrix-Vector Spaces. In *EMNLP*.

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

Swapna Somasundaran, Galileo Namata, Janyce Wiebe, and Lise Getoor. 2009. Supervised and unsupervised methods in employing discourse relations for improving opinion polarity classification. In *Proceedings of EMNLP*.

Radu Soricut and Daniel Marcu. 2003. Sentence Level Discourse Parsing using Syntactic and Lexical Information. In *NAACL*.

Rajen Subba and Barbara Di Eugenio. 2009. An effective Discourse Parser that uses Rich Linguistic Information. In *NAACL-HLT*, pages 566–574, Boulder, Colorado, June. Association for Computational Linguistics.

K. Sumita, K. Ono, T. Chino, T. Ukita, and S. Amano. 1992. A discourse structure analyzer for Japanese text. In *Proceedings International Conference on Fifth Generation Computer Systems*, pages 1133–1140.

Maite Taboada and William C Mann. 2006. Applications of rhetorical structure theory. *Discourse studies*, 8(4):567–588.

Benjamin Taskar, Carlos Guestrin, and Daphne Koller. 2003. Max-margin markov networks. In *NIPS*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word Representation: A Simple and General Method for Semi-Supervised Learning. In *Proceedings of ACL*, pages 384–394.

Tim Van de Cruys and Marianna Apidianaki. 2011. Latent Semantic Word Sense Induction and Disambiguation. In *Proceedings of ACL*, pages 1476–1485, Portland, Oregon, USA, June. Association for Computational Linguistics.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9:2759–2605, November.

Kimberly Voll and Maite Taboada. 2007. Not all words are created equal: Extracting semantic orientation as a function of adjective relevance. In *Proceedings of Australian Conference on Artificial Intelligence*.

Ngo Xuan Bach, Nguyen Le Minh, and Akira Shimazu. 2012. A Reranking Model for Discourse Segmentation using Subtree Features. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 160–168.

Chun-Nam John Yu and Thorsten Joachims. 2009. Learning structural SVMs with latent variables. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1169–1176. ACM.

Text-level Discourse Dependency Parsing

Sujian Li[1] **Liang Wang**[1] **Ziqiang Cao**[1] **Wenjie Li**[2]
[1] Key Laboratory of Computational Linguistics, Peking University, MOE, China
[2] Department of Computing, The Hong Kong Polytechnic University, HongKong
{lisujian,intfloat,ziqiangyeah}@pku.edu.cn
cswjli@comp.polyu.edu.hk

Abstract

Previous researches on Text-level discourse parsing mainly made use of constituency structure to parse the whole document into one discourse tree. In this paper, we present the limitations of constituency based discourse parsing and first propose to use dependency structure to directly represent the relations between elementary discourse units (EDUs). The state-of-the-art dependency parsing techniques, the Eisner algorithm and maximum spanning tree (MST) algorithm, are adopted to parse an optimal discourse dependency tree based on the arc-factored model and the large-margin learning techniques. Experiments show that our discourse dependency parsers achieve a competitive performance on text-level discourse parsing.

1 Introduction

It is widely agreed that no units of the text can be understood in isolation, but in relation to their context. Researches in discourse parsing aim to acquire such relations in text, which is fundamental to many natural language processing applications such as question answering, automatic summarization and so on.

One important issue behind discourse parsing is the representation of discourse structure. Rhetorical Structure Theory (RST) (Mann and Thompson, 1988), one of the most influential discourse theories, posits a hierarchical generative tree representation, as illustrated in Figure 1. The leaves of a tree correspond to contiguous text spans called Elementary Discourse Units (EDUs)[1]. The adjacent EDUs are combined into the larger text spans by rhetorical relations (e.g., Contrast and Elaboration) and the larger text spans continue to be combined until the whole text constitutes a parse tree. The text spans linked by rhetorical relations are annotated as either *nucleus* or *satellite* depending on how salient they are for interpretation. It is attractive and challenging to parse the whole text into one tree.

Since such a hierarchical discourse tree is analogous to a constituency based syntactic tree except that the constituents in the discourse trees are text spans, previous researches have explored different constituency based syntactic parsing techniques (eg. CKY and chart parsing) and various features (eg. length, position et al.) for discourse parsing (Soricut and Marcu, 2003; Joty et al., 2012; Reitter, 2003; LeThanh et al., 2004; Baldridge and Lascarides, 2005; Subba and Di Eugenio, 2009; Sagae, 2009; Hernault et al., 2010b; Feng and Hirst, 2012). However, the existing approaches suffer from at least one of the following three problems. First, it is difficult to design a set of production rules as in syntactic parsing, since there are no determinate generative rules for the interior text spans. Second, the different levels of discourse units (e.g. EDUs or larger text spans) occurring in the generative process are better represented with different features, and thus a uniform framework for discourse analysis is hard to develop. Third, to reduce the time complexity of the state-of-the-art constituency based parsing techniques, the approximate parsing approaches are prone to trap in local maximum.

In this paper, we propose to adopt the *dependency structure* in discourse representation to overcome the limitations mentioned above. Here is the basic idea: the discourse structure consists of EDUs which are linked by the binary, asymmetrical relations called *dependency relations*. A dependency relation holds between a subordinate EDU called the *dependent*, and another EDU on

[1] EDU segmentation is a relatively trivial step in discourse parsing. Since our work focus here is not EDU segmentation but discourse parsing. We assume EDUs are already known.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 25–35,
Baltimore, Maryland, USA, June 23-25 2014.

which it depends called the *head*, as illustrated in Figure 2. Each EDU has one head. So, the dependency structure can be seen as a set of *head-dependent* links, which are labeled by functional relations. Now, we can analyze the relations between EDUs directly, without worrying about any interior text spans. Since dependency trees contain much fewer nodes and on average they are simpler than constituency based trees, the current dependency parsers can have a relatively low computational complexity. Moreover, concerning linearization, it is well known that dependency structures can deal with non-projective relations, while constituency-based models need the addition of complex mechanisms like transformations, movements and so on. In our work, we adopt the graph based dependency parsing techniques learned from large sets of annotated dependency trees. The Eisner (1996) algorithm and maximum spanning tree (MST) algorithm are used respectively to parse the optimal projective and non-projective dependency trees with the large-margin learning technique (Crammer and Singer, 2003). To the best of our knowledge, we are the first to apply the dependency structure and introduce the dependency parsing techniques into discourse analysis.

The rest of this paper is organized as follows. Section 2 formally defines discourse dependency structure and introduces how to build a discourse dependency treebank from the existing RST corpus. Section 3 presents the discourse parsing approach based on the Eisner and MST algorithms. Section 4 elaborates on the large-margin learning technique as well as the features we use. Section 5 discusses the experimental results. Section 6 introduces the related work and Section 7 concludes the paper.

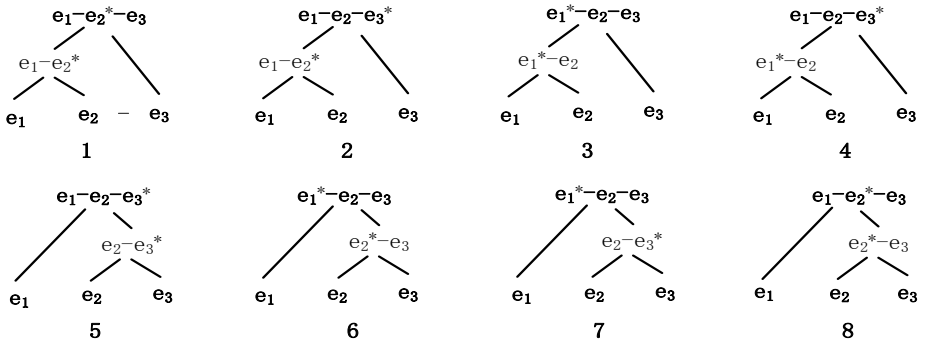


Figure 1: Headed Constituency based Discourse Tree Structure (e_1,e_2 and e_3 denote three EDUs, and * denotes the NUCLEUS constituent)

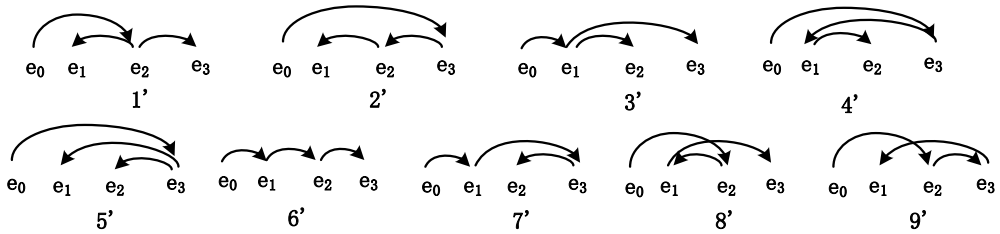


Figure 2: Discourse Dependency Tree Structures (e_1,e_2 and e_3 denote three EDUS, and the directed arcs denote one dependency relations. The artificial e_0 is also displayed here.)

2 Discourse Dependency Structure and Tree Bank

2.1 Discourse Dependency Structure

Similar to the syntactic dependency structure defined by McDonald (2005a, 2005b), we insert an artificial EDU e_0 in the beginning for each document and label the dependency relation linking from e_0 as **ROOT**. This treatment will simplify both formal definitions and computational implementations. Normally, we assume that each EDU should have one and only one head except for e_0. A labeled directed arc is used to represent the dependency relation from one head to its dependent. Then, discourse dependency structure can be formalized as the labeled directed graph, where nodes correspond to EDUs and labeled arcs correspond to labeled dependency relations.

We assume that the text[2] T is composed of n+1 EDUs including the artificial e_0. That is $T=e_0\ e_1\ e_2\ \dots\ e_n$. Let $R=\{r_1,r_2,\ \dots\ ,r_m\}$ denote a finite set of functional relations that hold between two EDUs. Then a discourse dependency graph can be denoted by $G=<V, A>$ where V denotes a set of nodes and A denotes a set of labeled directed arcs, such that for the text $T=e_0\ e_1\ e_2\ \dots\ e_n$ and the label set R the following holds:

(1) $V = \{ e_0, e_1, e_2, \dots e_n \}$

(2) $A \subseteq V \times R \times V$, where $<e_i, r, e_j> \in A$ represents an arc from the head e_i to the dependent e_j labeled with the relation r.

(3) If $<e_i, r, e_j> \in A$ then $<e_k, r', e_j> \notin A$ for all $k \neq i$

(4) If $<e_i, r, e_j> \in A$ then $<e_i, r', e_j> \notin A$ for all $r' \neq r$

The third condition assures that each EDU has one and only one head and the fourth tells that only one kind of dependency relation holds between two EDUs. According to the definition, we illustrate all the 9 possible unlabeled dependency trees for a text containing three EDUs in Figure 2. The dependency trees 1' to 7' are projective while 8' and 9' are non-projective with crossing arcs.

2.2 Our Discourse Dependency Treebank

To automatically conduct discourse dependency parsing, constructing a discourse dependency treebank is fundamental. It is costly to manually construct such a treebank from scratch. Fortunately, RST Discourse Treebank (RST-DT) (Carlson et al., 2001) is an available resource to help with.

A RST tree constitutes a hierarchical structure for one document through rhetorical relations. A total of 110 fine-grained relations (e.g. Elaboration-part-whole and List) were used for tagging RST-DT. They can be categorized into 18 classes (e.g. Elaboration and Joint). All these relations can be hypotactic ("mononuclear") or paratactic ("multi-nuclear"). A hypotactic relation holds between a *nucleus* span and an adjacent *satellite* span, while a paratactic relation connects two or more equally important adjacent *nucleus* spans. For convenience of computation, we convert the n-ary (n>2) RST trees[3] to binary trees through adding a new node for the latter n-1 nodes and assume each relation is connected to only one *nucleus*[4]. This departure from the original theory is not such a major step as it may appear, since any nucleus is known to contribute to the essential meaning. Now, each RST tree can be seen as a headed constituency based binary tree where the nuclei are heads and the children of each node are linearly ordered. Given three EDUs[5], Figure 1 shows the possible 8 headed constituency based trees where the superscript * denotes the heads (nuclei). We use dependency trees to simulate the headed constituency based trees.

Contrasting Figure 1 with Figure 2, we use dependency tree 1' to simulate binary trees 1 and 8, and dependency tress 2'- 7' to simulate binary trees 2-7 correspondingly. The rhetorical relations in RST trees are kept as the functional relations which link the two EDUs in dependency trees. With this kind of conversion, we can get our discourse dependency treebank. It is worth noting that the non-projective trees like 8' and 9' do not exist in our dependency treebank, though they are eligible according to the definition of discourse dependency graph.

3 Discourse Dependency Parsing

3.1 System Overview

As stated above, $T=e_0\ e_1\ \dots e_n$ represents an input text (document) where e_i denotes the i^{th} EDU of T. We use V to denote all the EDU nodes and $V\times R\times V_{-0}$ ($V_{-0}=V-\{e_0\}$) denote all the possible discourse dependency arcs. The goal of discourse dependency parsing is to parse an optimal spanning tree from $V\times R\times V_{-0}$. Here we follow the arc factored method and define the score of a dependency tree as the sum of the scores of all the arcs in the tree. Thus, the optimal dependency tree for T is a spanning tree with the highest score and obtained through the function $DT(T,\boldsymbol{w})$:

$$DT(T,\boldsymbol{w}) = argmax_{G_T \subseteq V\times R\times V_{-0}}\ score(T, G_T)$$
$$= argmax_{G_T \subseteq V\times R\times V_{-0}} \sum_{<e_i,r,e_j>\in G_T} \lambda(e_i, r, e_j)$$
$$= argmax_{G_T \subseteq V\times R\times V_{-0}} \sum_{<e_i,r,e_j>\in G_T} \boldsymbol{w}\cdot\mathbf{f}(e_i, r, e_j)$$

where G_T means a possible spanning tree with $score(T, G_T)$ and $\lambda(e_i, r, e_j)$ denotes the score of the arc $<e_i, r, e_j>$ which is calculated according to its feature representation $\mathbf{f}(e_i,r,e_j)$ and a weight vector $\boldsymbol{w}$.

Next, two basic problems need to be solved: how to find the dependency tree with the highest

[2] The two terms "text" and "document" are used interchangeably and represent the same meaning.

[3] According to our statistics, there are totally 381 n-ary relations in RST-DT.

[4] We set the first nucleus as the only nucleus.

[5] We can easily get all possible headed binary trees for one more complex text containing more than three EDUs, by extending the 8 possible situations for three EDUs.

score for T given all the arc scores (i.e. a parsing problem), and how to learn and compute the scores of arcs according to a set of arc features (i.e. a learning problem).

The following of this section addresses the first problem. Given the text T, we first reduce the multi-digraph composed of all possible arcs to the digraph. The digraph keeps only one arc $<e_i, r, e_j>$ between two nodes which satisfies $\lambda(e_i, r, e_j) = max_{r'}\lambda(e_i, r', e_j)$. Thus, we can proceed with a reduction from labeled parsing to unlabeled parsing. Next, two algorithms, i.e. the Eisner algorithm and MST algorithm, are presented to parse the projective and non-projective unlabeled dependency trees respectively.

3.2 Eisner Algorithm

It is well known that projective dependency parsing can be handled with the Eisner algorithm (1996) which is based on the bottom-up dynamic programming techniques with the time complexity of $O(n^3)$. The basic idea of the Eisner algorithm is to parse the left and right dependents of an EDU independently and combine them at a later stage. This reduces the overhead of indexing heads. Only two binary variables, i.e. c and d, are required to specify whether the heads occur leftmost or rightmost and whether an item is complete.

Eisner(T, λ)
Input: Text $T=e_0\, e_1 \ldots\, e_n$; Arc scores $\lambda(e_i,e_j)$
1 Instantiate E[i, i, d, c]=0.0 for all i, d, c
2 For m := 1 to n
3 For i := 1 to n
4 $j = i + m$
5 if $j > n$ then break;
6 # Create subgraphs with c=0 by adding arcs
7 E[i, j, 0, 0]=$\max_{i\le q<j}$ (E[i,q,1,1]+E[q+1,j,0,1]+$\lambda(e_j,e_i)$)
8 E[i, j, 1, 0]=$\max_{i\le q<j}$ (E[i,q,1,1]+E[q+1,j,0,1]+$\lambda(e_i,e_j)$)
9 # Add corresponding left/right subgraphs
10 E[i, j, 0, 1]=$\max_{i\le q<j}$ (E[i,q,0,1]+E[q,j,0,0]
11 E[i, j, 1, 1]=$\max_{i\le q<j}$ (E[i,q,1,0]+E[q,j,1,1])

Figure 3: Eisner Algorithm

Figure 3 shows the pseudo-code of the Eisner algorithm. A dynamic programming table E[i,j,d,c] is used to represent the highest scored subtree spanning e_i to e_j. d indicates whether e_i is the head (d=1) or e_j is head (d=0). c indicates whether the subtree will not take any more dependents (c=1) or it needs to be completed (c=0). The algorithm begins by initializing all length-one subtrees to a score of 0.0. In the inner loop, the first two steps (Lines 7 and 8) are to construct the new dependency arcs by taking the maximum over all the internal indices ($i \le q \le j$) in the span, and calculating the value of merging the two subtrees and adding one new arc. The last two steps (Lines 10 and 11) attempt to achieve an optimal left/right subtree in the span by adding the corresponding left/right subtree to the arcs that have been added previously. This algorithm considers all the possible subtrees. We can then get the optimal dependency tree with the score $E[0,n,1,1]$.

3.3 Maximum Spanning Tree Algorithm

As the bottom-up Eisner Algorithm must maintain the nested structural constraint, it cannot parse the non-projective dependency trees like 8' and 9' in Figure 2. However, the non-projective dependency does exist in real discourse. For example, the earlier text mainly talks about the topic A with mentioning the topic B, while the latter text gives a supplementary explanation for the topic B. This example can constitute a non-projective tree and its pictorial diagram is exhibited in Figure 4. Following the work of McDonald (2005b), we formalize discourse dependency parsing as searching for a maximum spanning tree (MST) in a directed graph.

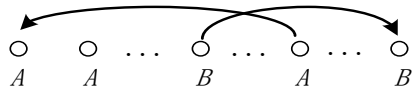


Figure 4: Pictorial Diagram of Non-projective Trees

Chu and Liu (1965) and Edmonds (1967) independently proposed the virtually identical algorithm named the Chu-Liu/Edmonds algorithm, for finding MSTs on directed graphs (McDonald et al. 2005b). Figure 5 shows the details of the Chu-Liu/Edmonds algorithm for discourse parsing. Each node in the graph greedily selects the incoming arc with the highest score. If one tree results, the algorithm ends. Otherwise, there must exist a cycle. The algorithm contracts the identified cycle into a single node and recalculates the scores of the arcs which go in and out of the cycle. Next, the algorithm recursively call itself on the contracted graph. Finally, those arcs which go in or out of one cycle will recover themselves to connect with the original nodes in V. Like McDonald et al. (2005b), we adopt an efficient implementation of the Chu-Liu/Edmonds algorithm that is proposed by Tarjan (1997) with $O(n^2)$ time complexity.

Chu-Liu-Edmonds(G, λ)
Input: Text $T=e_0\, e_1 \ldots\, e_n$; Arc scores $\lambda(e_i,e_j)$
A' = {<e_i, e_j>| e_i = argmax $\lambda(e_i,e_j)$; $1\le j\le|V|$}
G' = (V, A')
If G' has no cycles, then return G'
Find an arc set A_C that is a cycle in G'
<G_C, ep> = contract(G, A_C, λ)
G = (V, A)=Chu-Liu-Edmonds(G_C, λ)
For the arc <e_i,e_C> where ep(e_i,e_C)=e_j:
    A=$A\cup A_C\cup${<e_i,e_j>}-{<e_i,e_C>, <a(e_j),e_j>}
For the arc <e_C, e_i> where ep(e_C ,e_i)=e_j:
    A=$A\cup${<e_j,e_i>}-{<e_C,e_i>}
V = V
Return G
Contract(G=(V,A), A_C, λ)
Let G_C be the subgraph of G excluding nodes in C
Add a node e_C to G_C denoting the cycle C
For $e_j \in V$-C : $\exists e_i \in C$ <e_i,e_j>$\in A$
    Add arc <e_C,e_j> to G_C with
        ep(e_C,e_j)=argmax$_{e_i\in C}\lambda(e_i,e_j)$
    $\lambda(e_C,e_j)$ = λ(ep(e_C,e_j),e_j)
For $e_i \in V$-C: $\exists e_j \in$C (e_i,e_j)$\in$A
    Add arc <e_i,e_C> to G_C with
        ep(e_i,e_C)= =argmax$_{e_i\in C}$ [$\lambda(e_i,e_j)$-$\lambda(a(e_i),e_j)$]
    $\lambda(e_i,e_C)$ =$\lambda(e_i,e_j)$-$\lambda(a(e_i),e_j)$+score(C)
Return <G_C, ep>

Figure 5: Chu-Liu/Edmonds MST Algorithm

4 Learning

In Section 3, we assume that the arc scores are available. In fact, the score of each arc is calculated as a linear combination of feature weights. Thus, we need to determine the features for arc representation first. With referring to McDonald et al. (2005a; 2005b), we use the Margin Infused Relaxed Algorithm (MIRA) to learn the feature weights based on a training set of documents annotated with dependency structures $\{(T_i, \boldsymbol{y}_i)\}_{i=1}^{N}$ where $\boldsymbol{y}_i$ denotes the correct dependency tree for the text T_i.

4.1 Features

Following (Feng and Hirst, 2012; Lin et al., 2009; Hernault et al., 2010b), we explore the following 6 feature types combined with relations to represent each labeled arc <e_i, r, e_j> .

(1) WORD: The first one word, the last one word, and the first bigrams in each EDU, the pair of the two first words and the pair of the two last words in the two EDUs are extracted as features.

(2) POS: The first one and two POS tags in each EDU, and the pair of the two first POS tags in the two EDUs are extracted as features.

(3) Position: These features concern whether the two EDUs are included in the same sentence, and the positions where the two EDUs are located in one sentence, one paragraph, or one document.

(4) Length: The length of each EDU.

(5) Syntactic: POS tags of the dominating nodes as defined in Soricut and Marcu (2003) are extracted as features. We use the syntactic trees from the Penn Treebank to find the dominating nodes,.

(6) Semantic similarity: We compute the semantic relatedness between the two EDUs based on WordNet. The word pairs are extracted from (e_i, e_j) and their similarity is calculated. Then, we can get a weighted complete bipartite graph where words are deemed as nodes and similarity as weights. From this bipartite graph, we get the maximum weighted matching and use the averaged weight of the matches as the similarity between e_i and e_j. In particular, we use path_similarity, wup_similarity, res_similarity, jcn_similarity and lin_similarity provided by the nltk.wordnet.similarity (Bird et. al., 2009) package for calculating word similarity.

As for relations, we experiment two sets of relation labels from RST-DT. One is composed of 19 coarse-grained relations and the other 111 fine-grained relations[6].

4.2 MIRA based Learning

Margin Infused Relaxed Algorithm (MIRA) is an online algorithm for multiclass classification and is extended by Taskar et al. (2003) to cope with structured classification.

MIRA Input: a training set $\{(T_i, \boldsymbol{y}_i)\}_{i=1}^{N}$
$\boldsymbol{w}^0$ = 0; $\boldsymbol{v}$ = 0; j = 0
For *iter* := 1 to K
    For i := 1 to N
        update $\boldsymbol{w}$ according to $(T_i, \boldsymbol{y}_i)$:
        $\min \|\boldsymbol{w}^{j+1} - \boldsymbol{w}^{j}\|$
        s.t. $s(T_i, \boldsymbol{y}_i) - s(T_i, \boldsymbol{y}_i') \ge L(\boldsymbol{y}_i, \boldsymbol{y}_i')$
        where $\boldsymbol{y}_i' = DT(T_i, \boldsymbol{w}^j)$
        $\boldsymbol{v} = \boldsymbol{v} + \boldsymbol{w}^j$;
        $j = j+1$
$\boldsymbol{w} = \boldsymbol{v}/(K*N)$

Figure 6: MIRA based Learning

Figure 6 gives the pseudo-code of the MIRA algorithm (McDonld et al., 2005b). This algorithm is designed to update the parameters $\boldsymbol{w}$ using a single training instance $(T_i, \boldsymbol{y}_i)$ in each iteration. On each update, MIRA attempts to keep the norm of the change to the weight vector

[6] 19 relations include the original 18 relation in RST-DT plus one artificial **ROOT** relation. The 111 relations also include the ROOT relation.

as small as possible, which is subject to constructing the correct dependency tree under consideration with a margin at least as large as the loss of the incorrect dependency trees. We define the loss of a discourse dependency tree y_i' (denoted by $L(y_i, y_i')$) as the number of the EDUs that have incorrect heads. Since there are exponentially many possible incorrect dependency trees and thus exponentially many margin constraints, here we relax the optimization and stay with a single best dependency tree $y_i' = DT(T_i, w^j)$ which is parsed under the weight vector w^j. In this algorithm, the successive updated values of w are accumulated and averaged to avoid overfitting.

5 Experiments

5.1 Preparation

We test our methods experimentally using the discourse dependency treebank which is built as in Section 2. The training part of the corpus is composed of 342 documents and contains 18,765 EDUs, while the test part consists of 38 documents and 2,346 EDUs. The number of EDUs in each document ranges between 2 and 304. Two sets of relations are adopted. One is composed of 19 relations and Table 1 shows the number of each relation in the training and test corpus. The other is composed of 111 relations. Due to space limitation, Table 2 only lists the 10 highest-distributed relations with regard to their frequency in the training corpus.

The following experiments are conducted: (1) to measure the parsing performance with different relation sets and different feature types; (2) to compare our parsing methods with the state-of-the-art discourse parsing methods.

Relations	**Train**	**Test**	**Relations**	**Train**	**Test**
Elaboration	6879	796	Temporal	426	73
Attribution	2641	343	ROOT	342	38
Joint	1711	212	Compari.	273	29
Same-unit	1230	127	Condition	258	48
Contrast	944	146	Manner.	191	27
Explanation	849	110	Summary	188	32
Background	786	111	Topic-Cha.	187	13
Cause	785	82	Textual	147	9
Evaluation	502	80	TopicCom.	126	24
Enablement	500	46	**Total**	18765	2346

Table 1: Coarse-grained Relation Distribution

Relations	**Train**	**Test**
Elaboration-additional	2912	312
Attribution	2474	329
Elaboration-object-attribute-e	2274	250
List	1690	206
Same-unit	1230	127
Elaboration-additional-e	747	69
Circumstance	545	80
Explanation-argumentative	524	70
Purpose	430	43
Contrast	358	64

Table 2: 10 Highest Distributed Fine-grained Relations

5.2 Feature Influence on Two Relation Sets

So far, researches on discourse parsing avoid adopting too fine-grained relations and the relation sets containing around 20 labels are widely used. In our experiments, we observe that adopting a fine-grained relation set can even be helpful to building the discourse trees. Here, we conduct experiments on two relation sets that contain 19 and 111 labels respectively. At the same time, different feature types are tested their effects on discourse parsing.

Method	Features	Unlabeled Acc.	Labeled Acc.
Eisner	1+2	0.3602	0.2651
	1+2+3	0.7310	0.4855
	1+2+3+4	0.7370	0.4868
	1+2+3+4+5	**0.7447**	**0.4957**
	1+2+3+4+5+6	**0.7455**	**0.4983**
MST	1+2	0.1957	0.1479
	1+2+3	0.7246	0.4783
	1+2+3+4	0.7280	0.4795
	1+2+3+4+5	**0.7340**	**0.4915**
	1+2+3+4+5+6	**0.7331**	**0.4851**

Table 3: Performance Using Coarse-grained Relations.

Method	Feature types	Unlabeled Acc.	Labeled Acc.
Eisner	1+2	0.3743	0.2421
	1+2+3	0.7451	0.4079
	1+2+3+4	0.7472	0.4041
	1+2+3+4+5	**0.7506**	**0.4254**
	1+2+3+4+5+6	**0.7485**	**0.4288**
MST	1+2	0.2080	0.1300
	1+2+3	0.7366	0.4054
	1+2+3+4	0.7468	0.4071
	1+2+3+4+5	**0.7494**	**0.4288**
	1+2+3+4+5+6	**0.7460**	**0.4309**

Table 4: Performance Using Fine-grained Relations.

Based on the MIRA leaning algorithm, the Eisner algorithm and MST algorithm are used to parse the test documents respectively. Referring to the evaluation of syntactic dependency parsing,

we use *unlabeled accuracy* to calculate the ratio of EDUs that correctly identify their heads, *labeled accuracy* the ratio of EDUs that have both correct heads and correct relations. Table 3 and Table 4 show the performance on two relation sets. The numbers (1-6) represent the corresponding feature types described in Section 4.1.

From Table 3 and Table 4, we can see that the addition of more feature types, except the 6th feature type (semantic similarity), can promote the performance of relation labeling, whether using the coarse-grained 19 relations and the fine-grained 111 relations. As expected, the first and second types of features (WORD and POS) are the ones which play an important role in building and labeling the discourse dependency trees. These two types of features attain similar performance on two relation sets. The Eisner algorithm can achieve unlabeled accuracy around 0.36 and labeled accuracy around 0.26, while MST algorithm achieves unlabeled accuracy around 0.20 and labeled accuracy around 0.14.

The third feature type (Position) is also very helpful to discourse parsing. With the addition of this feature type, both unlabeled accuracy and labeled accuracy exhibit a marked increase. Especially, when applying MST algorithm on discourse parsing, unlabeled accuracy rises from around 0.20 to around 0.73. This result is consistent with Hernault's work (2010b) whose experiments have exhibited the usefulness of those position-related features. The other two types of features which are related to length and syntactic parsing, only promote the performance slightly.

As we employed the MIRA learning algorithm, it is possible to identify which specific features are useful, by looking at the weights learned to each feature using the training data. Table 5 selects 10 features with the highest weights in absolute value for the parser which uses the coarse-grained relations, while Table 6 selects the top 10 features for the parser using the fine-grained relations. Each row denotes one feature: the left part before the symbol "&" is from one of the 6 feature types and the right part denotes a specific relation. From Table 5 and Table 6, we can see that some features are reasonable. For example, The sixth feature in Table 5 represents that the dependency relation is preferred to be labeled *Explanation* with the fact that "because" is the first word of the dependent EDU. From these two tables, we also observe that most of the heavily weighted features are usually related to those highly distributed relations. When using the coarse-grained relations, the popular relations (eg. Elaboration, Attribution and Joint) are always preferred to be labeled. When using the fine-grained relations, the large relations including List and Elaboration-object-attribute-e are given the precedence of labeling. This phenomenon is mainly caused by the sparseness of the training corpus and the imbalance of relations. To solve this problem, the augment of training corpus is necessary.

	Feature description	**Weight**
1	Last two words in dependent EDU are "appeals court" & Joint	0.475
2	First word in dependent EDU is "racked" & Elaboration	0.445
3	First two words in head EDU are "I 'd" & Attribution	0.324
4	Last word in dependent EDU is "in" & Elaboration	-0.323
5	The res_similarity between two EDUs is 0 & Elaboration	0.322
6	First word in dependent EDU is "because" & Explanation	0.306
7	First POS in head EDU is "DT" & Joint	-0.299
8	First two words in dependent EDU are "that required" & Elaboration	0.287
9	First two words in dependent EDU are "that the" & Elaboration	0.277
10	First word in dependent EDU is "because" & Cause	0.265

Table 5: Top 10 Feature Weights for Coarse-grained Relation Labeling (Eisner Algorithm)

	Features	**Weight**
1	Last two words in dependent EDU are "appeals court" & List	0.576
2	First two words in head EDU are "I 'd" & Attribution	0.385
3	First two words in dependent EDU is "that the" & Elaboration-object-attribute-e	0.348
4	First POS in head EDU is "DT" & List	-0.323
5	Last word in dependent EDU is "in" & List	-0.286
6	First word in dependent EDU is "racked" & Elaboration-object-attribute-e	0.445
7	First two word pairs are <"In an","But even"> & List	-0.252
8	Dependent EDU has a dominating node tagged "CD"& Elaboration-object-attribute-e	-0.244
9	First two words in dependent EDU are "patents disputes" & Purpose	0.231
10	First word in dependent EDU is "to" & Purpose	0.230

Table 6: Top 10 Feature Weights for Coarse-grained Relation Labeling (Eisner Algorithm)

Unlike previous discourse parsing approaches, our methods combine tree building and relation labeling into a uniform framework naturally. This means that relations play a role in building the dependency tree structure. From Table 3 and Table 4, we can see that fine-grained relations are more helpful to building unlabeled discourse

trees more than the coarse-grained relations. The best result of unlabeled accuracy using 111 relations is 0.7506, better than the best performance (0.7447) using 19 relations. We can also see that the labeled accuracy using the fine-grained relations can achieve 0.4309, only 0.06 lower than the best labeled accuracy (0.4915) using the coarse-grained relations.

In addition, comparing the MST algorithm with the Eisner algorithm, Table 3 and Table 4 show that their performances are not significantly different from each other. But we think that MST algorithm has more potential in discourse dependency parsing, because our converted discourse dependency treebank contains only projective trees and somewhat suppresses the MST algorithm to exhibit its advantage of parsing non-projective trees. In fact, we observe that some non-projective dependencies produced by the MST algorithm are even reasonable than what they are in the dependency treebank. Thus, it is important to build a manually labeled discourse dependency treebank, which will be our future work.

5.3 Comparison with Other Systems

The state-of-the-art discourse parsing methods normally produce the constituency based discourse trees. To comprehensively evaluate the performance of a labeled constituency tree, the blank tree structure ('S'), the tree structure with nuclearity indication ('N'), and the tree structure with rhetorical relation indication but no nuclearity indication ('R') are evaluated respectively using the *F* measure (Marcu 2000).

To compare our discourse parsers with others, we adopt MIRA and Eisner algorithm to conduct discourse parsing with all the 6 types of features and then convert the produced projective dependency trees to constituency based trees through their correspondence as stated in Section 2. Our parsers using two relation sets are named *Our-coarse* and *Our-fine* respectively. The inputted EDUs of our parsers are from the standard segmentation of RST-DT. Other text-level discourse parsing methods include: (1) *Percep-coarse*: we replace MIRA with the averaged perceptron learning algorithm and the other settings are the same with *Our-coarse*; (2) *HILDA-manual* and *HILDA-seg* are from Hernault (2010b)'s work, and their inputted EDUs are from RST-DT and their own EDU segmenter respectively; (3) *LeThanh* indicates the results given by LeThanh el al. (2004), which built a multi-level rule based parser and used 14 relations evaluated on 21 documents from RST-DT; (4) *Marcu* denotes the results given by Marcu(2000)'s decision-tree based parser which used 15 relations evaluated on unspecified documents.

Table 7 shows the performance comparison for all the parsers mentioned above. *Human* denotes the manual agreement between two human annotators. From this table, we can see that both our parsers perform better than all the other parsers as a whole, though our parsers are not developed directly for constituency based trees. Our parsers do not exhibit obvious advantage than *HILDA-manual* on labeling the blank tree structure, because our parsers and *HILDA-manual* all perform over 94% of *Human* and this performance level somewhat reaches a bottleneck to promote more. However, our parsers outperform the other parsers on both nuclearity and relation labeling. *Our-coarse* achieves 94.2% and 91.8% of the human *F*-scores, on labeling nuclearity and relation respectively, while *Our-fine* achieves 95.2% and 87.6%. We can also see that the averaged perceptron learning algorithm, though simple, can achieve a comparable performance, better than *HILDA-manual*. The parsers *HILDA-seg*, *LeThanh* and *Marcu* use their own automatic EDU segmenters and exhibit a relatively low performance. This means that EDU segmentation is important to a practical discourse parser and worth further investigation.

	S	N	R
Our-coarse	**82.9**	**73.0**	**60.6**
Our-fine	**83.4**	**73.8**	**57.8**
Percep-coarse	82.3	72.6	59.4
HILDA-manual	83.0	68.4	55.3
HILDA-seg	72.3	59.1	47.8
LeThanh	53.7	47.1	39.9
Marcu	44.8	30.9	18.8
Human	**88.1**	**77.5**	**66.0**

Table 7: Full Parser Evaluation

	MAFS	WAFS	Acc
Our-coarse	**0.454**	**0.643**	**66.84**
Percep-coarse	0.438	0.633	65.37
Feng	0.440	0.607	65.30
HILDA-manual	0.428	0.604	64.18
Baseline	-	-	35.82

Table 8: Relation Labeling Performance

To further compare the performance of relation labeling, we follow Hernault el al. (2010a) and use Macro-averaged F-score (MAFS) to evaluate each relation. Due to space limitation, we do not list the *F* scores for each relation. Macro-averaged F-score is not influenced by the number of instances that are contained in each

relation. Weight-averaged F-score (WAFS) weights the performance of each relation by the number of its existing instances. Table 8 compares our parser *Our-coarse* with other parsers *HILDA-manual*, *Feng* (Feng and Hirst, 2012) and *Baseline*. *Feng* (Feng and Hirst, 2012) can be seen as a strengthened version of HILDA which adopts more features and conducts feature selection. *Baseline* always picks the most frequent relation (i.e. Elaboration). From the results, we find that *Our-coarse* consistently provides superior performance for most relations over other parsers, and therefore results in higher MAFS and WAFS.

6 Related Work

So far, the existing discourse parsing techniques are mainly based on two well-known treebanks. One is the Penn Discourse TreeBank (PDTB) (Prasad et al., 2007) and the other is RST-DT.

PDTB adopts the predicate-arguments representation by taking an implicit/explicit connective as a predication of two adjacent sentences (arguments). Then the discourse relation between each pair of sentences is annotated independently to characterize its predication. A majority of researches regard discourse parsing as a classification task and mainly focus on exploiting various linguistic features and classifiers when using PDTB (Wellner et al., 2006; Pitler et al., 2009; Wang et al., 2010). However, the predicate-arguments annotation scheme itself has such a limitation that one can only obtain the local discourse relations without knowing the rich context.

In contrast, RST and its treebank enable people to derive a complete representation of the whole discourse. Researches have begun to investigate how to construct a RST tree for the given text. Since the RST tree is similar to the constituency based syntactic tree except that the constituent nodes are different, the syntactic parsing techniques have been borrowed for discourse parsing (Soricut and Marcu, 2003; Baldridge and Lascarides, 2005; Sagae, 2009; Hernault et al., 2010b; Feng and Hirst, 2012). Soricut and Marcu (2003) use a standard bottom-up chart parsing algorithm to determine the discourse structure of sentences. Baldridge and Lascarides (2005) model the process of discourse parsing with the probabilistic head driven parsing techniques. Sagae (2009) apply a transition based constituent parsing approach to construct a RST tree for a document. Hernault et al. (2010b) develop a greedy bottom-up tree building strategy for discourse parsing. The two adjacent text spans with the closest relations are combined in each iteration. As the extension of Hernault's work, Feng and Hirst (2012) further explore various features aiming to achieve better performance. However, as analyzed in Section 1, there exist three limitations with the constituency based discourse representation and parsing. We innovatively adopt the dependency structure, which can be benefited from the existing RST-DT, to represent the discourse. To the best of our knowledge, this work is the first to apply dependency structure and dependency parsing techniques in discourse analysis.

7 Conclusions

In this paper, we present the benefits and feasibility of applying dependency structure in text-level discourse parsing. Through the correspondence between constituency-based trees and dependency trees, we build a discourse dependency treebank by converting the existing RST-DT. Based on dependency structure, we are able to directly analyze the relations between the EDUs without worrying about the additional interior text spans, and apply the existing state-of-the-art dependency parsing techniques which have a relatively low time complexity. In our work, we use the graph based dependency parsing techniques learned from the annotated dependency trees. The Eisner algorithm and the MST algorithm are applied to parse the optimal projective and non-projective dependency trees respectively based on the arc-factored model. To calculate the score for each arc, six types of features are explored to represent the arcs and the feature weights are learned based on the MIRA learning technique. Experimental results exhibit the effectiveness of the proposed approaches. In the future, we will focus on non-projective discourse dependency parsing and explore more effective features.

Acknowledgments

This work was partially supported by National High Technology Research and Development Program of China (No. 2012AA011101), National Key Basic Research Program of China (No. 2014CB340504), National Natural Science Foundation of China (No. 61273278), and National Key Technology R&D Program (No: 2011BAH10B04-03). We also thank the three anonymous reviewers for their helpful comments.

References

Jason Baldridge and Alex Lascarides. 2005. *Probabilistic Head-driven Parsing for Discourse Structure*. In Proceedings of the Ninth Conference on Computational Natural Language Learning, pages 96–103.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python — Analyzing Text with the Natural Language Toolkit*. O'Reilly.

Lynn Carlson, Daniel Marcu, and Mary E. Okurowski. 2001. *Building a Discourse-tagged Corpus in the Framework of Rhetorical Structure Theory*. Proceedings of the Second SIGdial Workshop on Discourse and Dialogue-Volume 16, pages 1–10.

Yoeng-Jin Chu and Tseng-Hong Liu. 1965. *On the Shortest Arborescence of a Directed Graph*, Science Sinica, v.14, pp.1396-1400.

Koby Crammer and Yoram Singer. 2003. *Ultraconservative Online Algorithms for Multiclass Problems*. JMLR.

Jack Edmonds. 1967. *Optimum Branchings*, J. Research of the National Bureau of Standards, 71B, pp.233-240.

Jason Eisner. 1996. *Three New Probabilistic Models for Dependency Parsing: An Exploration*. In Proc. COLING.

Vanessa Wei Feng and Graeme Hirst. *Text-level Discourse Parsing with Rich Linguistic Features*, Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pages 60–68, Jeju, Republic of Korea, 8-14 July 2012.

Hugo Hernault, Danushka Bollegala, and Mitsuru Ishizuka. 2010a. *A Semi-supervised Approach to Improve Classification of Infrequent Discourse Relations Using Feature Vector Extension*. In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 399–409, Cambridge, MA, October. Association for Computational Linguistics.

Hugo Hernault, Helmut Prendinger, David A. duVerle, and Mitsuru Ishizuka. 2010b. *HILDA: A Discourse Parser Using Support Vector Machine Classification*. Dialogue and Discourse, 1(3):1–33.

Shafiq Joty, Giuseppe Carenini and Raymond T. Ng. *A Novel Discriminative Framework for Sentence-level Discourse Analysis*. EMNLP-CoNLL '12 Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning Stroudsburg, PA, USA.

Huong LeThanh, Geetha Abeysinghe, and Christian Huyck. 2004. *Generating Discourse Structures for Written Texts*. In Proceedings of the 20th International Conference on Computational Linguistics, pages 329– 335.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. *Recognizing Implicit Discourse Relations in the Penn Discourse Treebank*. In Proceedings of the 2009 Conference on Empirical Method in Natural Language Processing, Vol. 1, EMNLP'09, pages 343-351.

William Mann and Sandra Thompson. 1988. *Rhetorical Structure Theory: Toward a Functional Theory of Text Organization*. Text, 8(3):243–281.

Daniel Marcu. 2000. *The Theory and Practice of Discourse Parsing and Summarization*. MIT Press, Cambridge, MA, USA.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. *Online Large-Margin Training of Dependency Parsers*, 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005) .

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005b. *Non-projective Dependency Parsing using Spanning Tree Algorithms*, Proceedings of HLT/EMNLP 2005.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. *Automatic Sense Prediction for Implicit Discourse Relations in Text*, In Proc. of the 47th ACL. pages 683-691.

Rashmi Prasad, Eleni Miltsakaki, Nikhil Dinesh, Alan Lee, Aravind Joshi, Livio Robaldo, and Bonnie Webber. 2007. *The Penn Discourse Treebank 2.0 Annotation Manual*. The PDTB Research Group, December.

David Reitter. 2003. *Simple Signals for Complex Rhetorics: On Rhetorical Analysis with Rich-feature Support Vector Models*. LDV Forum, 18(1/2):38–52.

Kenji Sagae. 2009. *Analysis of discourse structure with syntactic dependencies and data-driven shift-reduce parsing*. In Proceedings of the 11th International Conference on Parsing Technologies, pages 81-84.

Radu Soricut and Daniel Marcu. 2003. *Sentence level discourse parsing using syntactic and lexical information*. In Proceedings of the 2003 Conference

of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, Volume 1, pages 149–156.

Rajen Subba and Barbara Di Eugenio. 2009. *An effective discourse parser that uses rich linguistic information*. In Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pages 566–574.

Robert Endre Tarjan, 1977. *Finding Optimum Branchings*, *Networks*, v.7, pp.25-35.

Ben Taskar, Carlos Guestrin and Daphne Koller. 2003. *Max-margin Markov Networks*. In Proc. NIPS.

Bonnie Webber. 2004. *D-LTAG: Extending Lexicalized TAG to Discourse*. Cognitive Science, 28(5):751–779.

Wen Ting Wang, Jian Su and Chew Lim Tan. 2010. *Kernel based Discourse Relation Recognition with Temporal Ordering Information*, In Proc. of ACL'10. pages 710-719.

Ben Wellner, James Pustejovsky, Catherine Havasi, Anna Rumshisky and Roser Sauri. 2006. *Classification of Discourse Coherence Relations: an Exploratory Study Using Multiple Knowledge Sources*. In Proc.of the 7th SIGDIAL Workshop on Discourse and Dialogue. pages 117-125.

Discovering Latent Structure in Task-Oriented Dialogues

Ke Zhai[*]
Computer Science, University of Maryland
College Park, MD 20740
zhaike@cs.umd.edu

Jason D. Williams
Microsoft Research
Redmond, WA 98052
jason.williams@microsoft.com

Abstract

A key challenge for computational conversation models is to discover latent structure in task-oriented dialogue, since it provides a basis for analysing, evaluating, and building conversational systems. We propose three new unsupervised models to discover latent structures in task-oriented dialogues. Our methods synthesize hidden Markov models (for underlying state) and topic models (to connect words to states). We apply them to two real, non-trivial datasets: human-computer spoken dialogues in bus query service, and human-human text-based chats from a live technical support service. We show that our models extract meaningful state representations and dialogue structures consistent with human annotations. Quantitatively, we show our models achieve superior performance on held-out log likelihood evaluation and an ordering task.

1 Introduction

Modeling human conversation is a fundamental scientific pursuit. In addition to yielding basic insights into human communication, computational models of conversation underpin a host of real-world applications, including interactive dialogue systems (Young, 2006), dialogue summarization (Murray et al., 2005; Daumé III and Marcu, 2006; Liu et al., 2010), and even medical applications such as diagnosis of psychological conditions (DeVault et al., 2013).

Computational models of conversation can be broadly divided into two genres: *modeling* and *control*. *Control* is concerned with choosing actions in interactive settings—for example to maximize task completion—using reinforcement learning (Levin et al., 2000), supervised learning (Hurtado et al., 2010), hand-crafted rules (Larsson and Traum, 2000), or mixtures of these (Henderson and Lemon, 2008). By contrast, *modeling*—the genre of this paper—is concerned with inferring a phenomena in an existing corpus, such as dialogue acts in two-party conversations (Stolcke et al., 2000) or topic shifts in multi-party dialogues (Galley et al., 2003; Purver et al., 2006; Hsueh et al., 2006; Banerjee and Rudnicky, 2006).

Many past works rely on supervised learning or human annotations, which usually requires manual labels and annotation guidelines (Jurafsky et al., 1997). It constrains scaling the size of training examples, and application domains. By contrast, unsupervised methods operate only on the observable signal (e.g. words) and are estimated without labels or their attendant limitations (Crook et al., 2009). They are particularly relevant because conversation is a *temporal process* where models are trained to infer a latent *state* which evolves as the dialogue progresses (Bangalore et al., 2006; Traum and Larsson, 2003).

Our basic approach is to assume that each utterance in the conversation is in a latent *state*, which has a causal effect on the words the conversants produce. Inferring this model yields basic insights into the structure of conversation and also has broad practical benefits, for example, speech recognition (Williams and Balakrishnan, 2009), natural language generation (Rieser and Lemon, 2010), and new features for dialogue policy optimization (Singh et al., 2002; Young, 2006).

There has been limited past work on unsupervised methods for conversation modeling. Chotimongkol (2008) studies task-oriented conversation and proposed a model based on a *hidden Markov model* (HMM). Ritter et al. (2010) extends it by introducing additional word sources, and applies to *non-task-oriented* conversations—social interactions on Twitter, where the subjects

[*]Work done at Microsoft Research.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 36–46,
Baltimore, Maryland, USA, June 23-25 2014.

discussed are very diffuse. The additional word sources capture the subjects, leaving the state-specific models to express common dialogue flows such as question/answer pairs.

In this paper, we retain the underlying HMM, but assume words are emitted using *topic models* (TM), exemplified by *latent Dirichlet allocation* (Blei et al., 2003, LDA). LDA assumes each word in an utterance is drawn from one of a set of latent topics, where each topic is a multinomial distribution over the vocabulary. The key idea is that the set of topics is *shared* across all states, and each state corresponds to a mixture of topics. We propose three model variants that link topics and states in different ways.

Sharing topics across states is an attractive property in task-oriented dialogue, where a single concept can be discussed at many points in a dialogue, yet different topics often appear in predictable sequences. Compared to past works, the *decoupling* of states and topics gives our models more expressive power and the potential to be more data efficient. Empirically, we find that our models outperform past approaches on two real-world corpora of task-oriented dialogues.

This paper is organized as follows: Section 2 introduces two task-oriented domains and corpora; Section 3 details three new unsupervised generative models which combine HMMs and LDA and efficient inference schemes; Section 4 evaluates our models qualitatively and quantitatively, and finally conclude in Section 5.

2 Data

To test the generality of our models, we study two very different datasets: a set of human-computer spoken dialogues in quering bus timetable (*BusTime*), and a set of human-human text-based dialogues in the technical support domain (*TechSupport*). In *BusTime*, the conversational structure is known because the computer followed a deterministic program (Williams, 2012), making it possible to directly compare an inferred model to ground truth on this corpus.[1] In *TechSupport*, there is no known flowchart,[2] making this a realistic application of unsupervised methods.

[1] Available for download at `http://research.microsoft.com/en-us/events/dstc/`

[2] Technical support human agents use many types of documentation—mainly checklists and guidelines, but in general, there are no flowcharts.

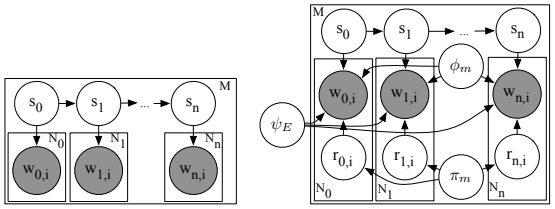
BusTime This corpus consists of logs of telephone calls between a spoken dialogue system and real bus users in Pittsburgh, USA (Black et al., 2010). For the user side, the words logged are the words recognized by the automatic speech recognizer. The vocabulary of the recognizer was constrained to the bus timetable task, so only words known to the recognizer in advance are output. Even so, the word error rate is approximately 30-40%, due to the challenging audio conditions of usage—with traffic noise and extraneous speech. The system asked users sequentially for a bus route, origin and destination, and optionally date and time. The system confirmed low-confidence speech recognition results. Due to the speech recognition channel, system and user turns always alternate. An example dialogue is given below:

System: Say a route like ⟨bus-route⟩, or say I'm not sure.
User: ⟨bus-route⟩.
System: I thought you said ⟨bus-route⟩, is that right?
User: Yes.
System: Say where're you leaving from, like ⟨location⟩.
User: ⟨location⟩.
System: Okay, ⟨location⟩, where are you going to?
...

We discard dialogues with fewer than 20 utterances. We also map all named entities (e.g., "downtown" and "28X") to their semantic types (resp. ⟨location⟩ and ⟨bus-route⟩) to reduce vocabulary size. The corpus we use consists of approximately 850 dialogue sessions or 30,000 utterances. It contains 370,000 tokens (words or semantic types) with vocabulary size 250.

TechSupport This corpus consists of logs of real web-based human-human text "chat" conversations between clients and technical support agents at a large corporation. Usually, clients and agents first exchange names and contact information; after that, dialogues are quite free-form, as agents ask questions and suggest fixes. Most dialogues ultimately end when the client's issue has been resolved; some clients are provided with a reference number for future follow-up. An example dialogue is given below:

Agent: Welcome to the answer desk! My name is ⟨agent-name⟩. How can I help you today?
Agent: May I have your name, email and phone no.?
Client: Hi, ⟨agent-name⟩. I recently installed new software but I kept getting error, can you help me?
Agent: Sorry to hear that. Let me help you with that.
Agent: May I have your name, email and phone no.?
Client: The error code is ⟨error-code⟩.
Client: It appears every time when I launch it.
Client: Sure. My name is ⟨client-name⟩.
Client: My email and phone are ⟨email⟩, ⟨phone⟩.
Agent: Thanks, ⟨client-name⟩, please give me a minute.



(a) *LM-HMM* (b) *LM-HMMS*

Figure 1: Plate diagrams of baseline models, from existing work (Chotimongkol, 2008; Ritter et al., 2010). Variable definitions are given in the text.

...

This data is less structured than *BusTime*; clients' issues span software, hardware, networking, and other topics. In addition, clients use common internet short-hand (e.g., "thx", "gtg", "ppl", "hv", etc), with mis-spellings (e.g., "ofice", "offfice", "erorr", etc). In addition, chats from the web interface are segmented into turns when a user hits "Enter" on a keyboard. Therefore, clients' input and agents' responses do **not** necessarily alternate consecutively, e.g., an agent's response may take multiple turns as in the above example. Also, it is unreasonable to group consecutive chats from the same party to form a "alternating" structure like *BusTime* dataset due to the asynchronism of different states. For instance, the second block of client inputs clearly comes from two different states which should *not* be merged together.

We discard dialogues with fewer than 30 utterances. We map named entities to their semantic types, apply stemming, and remove stop words.[3] The corpus we use contains approximately $2,000$ dialogue sessions or $80,000$ conversation utterances. It consists of $770,000$ tokens, with a a vocabulary size of $6,600$.

3 Latent Structure in Dialogues

In this work, our goal is to infer latent structure presented in task-oriented conversation. We assume that the structure can be encoded in a probabilistic state transition diagram, where the dialogue is in one state at each utterance, and states have a causal effect on the words observed. We assume the boundaries between utterances are given, which is trivial in many corpora.

The simplest formulation we consider is an HMM where each state contains a unigram *language model* (LM), proposed by Chotimongkol (2008) for task-oriented dialogue and originally developed for discourse analysis by Barzilay and Lee (2004). We call it *LM-HMM* as in Figure 1(a). For a corpus of M dialogues, the m-th dialogue contains n utterances, each of which contains N_n words (we omit index m from terms because it will be clear from context). At n-th utterance, we assume the dialogue is in some latent state s_n. Words in n-th utterance $w_{n,1}, \ldots, w_{n,N_n}$ are generated (independently) according to the *LM*. When an utterance is complete, the next state is drawn according to *HMM*, i.e., $P(s'|s)$.

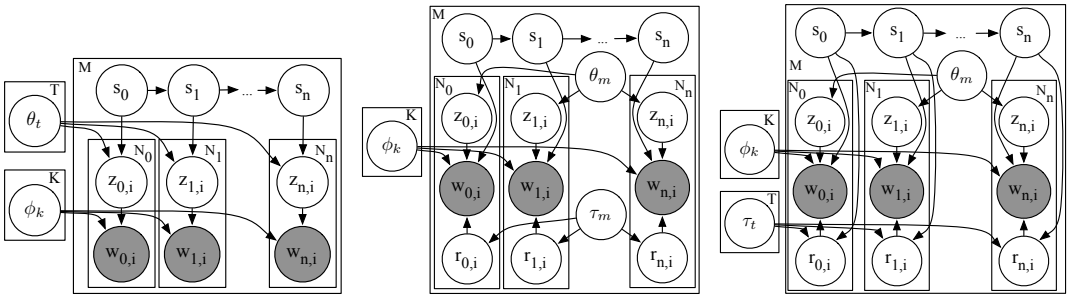
While *LM-HMM* captures the basic intuition of conversation structure, it assumes words are conditioned only on state. Ritter et al. (2010) extends *LM-HMM* to allow words to be emitted from two *additional* sources: the topic of current dialogue $\boldsymbol{\phi}$, or a background *LM* $\boldsymbol{\psi}$ shared across all dialogues. A multinomial $\boldsymbol{\pi}$ indicates the expected fraction of words from these three sources. For every word in an utterance, first draw a source indicator r from $\boldsymbol{\pi}$, and then generate the word from the corresponding source. We call it *LM-HMMS* (Figure 1(b)). Ritter et al. (2010) finds these alternate sources are important in non-task-oriented domains, where events are diffuse and fleeting. For example, Twitter exchanges often focus on a particular event (labeled X), and follow patterns like "saw X last night?", "X was amazing". Here X appears throughout the dialogue but does not help to distinguish conversational states in social media. We also explore similar variants.

In this paper, these two models form our baselines. For all models, we use *Markov chain Monte Carlo* (MCMC) inference (Neal, 2000) to find latent variables that best fit observed data. We also assume symmetric Dirichlet priors on all multinomial distributions and apply collapsed Gibbs sampling. In the rest of this section, we present our models and their inference algorithms in turn.

3.1 TM-HMM

Our approach is to modify the emission probabilities of states to be *distributions over topics* rather than distributions over words. In other words, instead of generating words via a *LM*, we generate words from a *topic model* (TM), where each state maps to a mixture of topics. The key benefit of this additional layer of abstraction is to enable states to express higher-level concepts through pooling of topics across states. For example, topics might be inferred for content like "bus-route" or "lo-

[3] We used regular expression to map named entities, and Porter stemmer in NLTK to stem all tokens.



(a) *TM-HMM* (b) *TM-HMMS* (c) *TM-HMMSS*

Figure 2: Plate diagrams of proposed models. *TM-HMM* is an HMM with state-wise topic distributions. *TM-HMMS* adds session-wise topic distribution and a source generator. *TM-HMMSS* adds a state-wise source generator. Variable definitions are given in the text.

cations"; and other topics for dialogue acts, like to "ask" or "confirm" information. States could then be *combinations* of these, e.g., a state might express "ask bus route" or "confirm location". This approach also decouples the number of topics from the number of states. Throughout this paper, we denote the number of topics as K and the number of states as T. We index words, turns and dialogues in the same ways as baseline models.

We develop three generative models. In the first variant (*TM-HMM*, Figure 2(a)), we assume every state s in HMM is associated with a distribution over topics $\boldsymbol{\theta}$, and topics generate words $\boldsymbol{w}$ at each utterance. The other two models allow words to be generated from different sources (in addition to states), akin to the *LM-HMMS* model.

TM-HMM generates a dialogue as following:

1: For each utterance n in that dialogue, sample a state s_n based on the previous state s_{n-1}.
2: For each word in utterance n, first draw a topic z from the state-specified distribution over topics $\boldsymbol{\theta}_{s_n}$ conditioned on s_n, then generate word w from the topic-specified distribution over vocabulary $\boldsymbol{\phi}_z$ based on z.

We assume $\boldsymbol{\theta}$'s and $\boldsymbol{\phi}$'s are drawn from corresponding Dirichlet priors, as in *LDA*.

The posterior distributions of state assignment s_n and topic assignment $z_{n,i}$ are

$$p(s_n|\boldsymbol{s}_{-n}, \boldsymbol{z}, \boldsymbol{\alpha}, \boldsymbol{\gamma}) \propto p(s_n|\boldsymbol{s}_{-n}, \boldsymbol{\gamma}) \cdot p(\boldsymbol{z}_n|\boldsymbol{s}, \boldsymbol{z}_{-n}, \boldsymbol{\alpha}), \quad (1)$$

$$p(z_{n,i}|\boldsymbol{s}, \boldsymbol{w}, \boldsymbol{z}_{-(n,i)}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \propto p(z_{n,i}|\boldsymbol{s}, \boldsymbol{z}_{-(n,i)}, \boldsymbol{\alpha}) \cdot p(w_{n,i}|s_n, \boldsymbol{w}_{-(n,i)}, \boldsymbol{z}, \boldsymbol{\beta}),$$

where $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$ are symmetric Dirichlet priors on state-wise topic distribution $\boldsymbol{\theta}_t$'s, topic-wise word distribution $\boldsymbol{\phi}_t$'s and state transition multinomials, respectively. All probabilities can be computed using collapsed Gibbs sampler for *LDA* (Griffiths and Steyvers, 2004) and *HMM* (Goldwater and Griffiths, 2007). We iteratively sample all parameters until convergence.

3.2 TM-HMMS

TM-HMMS (Figure 2(b)) extends *TM-HMM* to allow words to be generated either from state *LM* (as in *LM-HMM*), or a set of dialogue topics (akin to *LM-HMMS*). Because task-oriented dialogues usually focus on a specific domain, a set of words appears repeatedly throughout a given dialogue. Therefore, the topic distribution is often stable throughout the entire dialogue, and does not vary from turn to turn. For example, in the troubleshooting domain, dialogues about network connections, desktop productivity, and anti-virus software could each map to different session-wide topics. To express this, words in the *TM-HMMS* model are generated either from a dialogue-specific topic distribution, or from a state-specific language model.[4] A distribution over sources is sampled once at the beginning of each dialogue and selects the expected fraction of words generated from different sources.

The generative story for a dialogue session is:

1: At the beginning of each session, draw a distribution over topics $\boldsymbol{\theta}$ and a distribution over word sources $\boldsymbol{\tau}$.
2: For each utterance n in the conversation, draw a state s_n based on previous state s_{n-1}.
3: For each word in utterance n, first choose a word source r according to $\boldsymbol{\tau}$, and then depending on r, generate a word w either from the session-wide topic distribution $\boldsymbol{\theta}$ or the language model specified by the state s_n.

[4]Note that a *TM-HMMS* model with state-specific topic models (instead of state-specific language models) would be subsumed by *TM-HMM*, since one topic could be used as the background topic in *TM-HMMS*.

Again, we impose Dirichlet priors on distributions over topics $\boldsymbol{\theta}$'s and distributions over words $\boldsymbol{\phi}$'s as in *LDA*. We also assume the distributions over sources $\boldsymbol{\tau}$'s are governed by a Beta distribution.

The session-wide topics is slightly different from that used in *LM-HMMS*: *LM-HMMS* was developed for social chats on Twitter where topics are very diffuse and unlikely to repeat; hence often unique to each dialogue. By contrast, our models are designed for task-oriented dialogues which pertain to a given domain where topics are more tightly clustered; thus, in *TM-HMMS* session-wide topics are shared across the corpus.

The posterior distributions of state assignment s_n, word source $r_{n,i}$ and topic assignment $z_{n,i}$ are

$$\begin{aligned}
p(s_n|\boldsymbol{r}, \boldsymbol{s}_{-n}, \boldsymbol{w}, \boldsymbol{\gamma}, \boldsymbol{\pi}) &\propto p(s_n|\boldsymbol{s}_{-n}, \boldsymbol{\gamma}) \\
&\quad \cdot p(\boldsymbol{w}_n|\boldsymbol{r}, \boldsymbol{s}, \boldsymbol{\pi}), \\
p(r_{n,i}|\boldsymbol{r}_{-(n,i)}, \boldsymbol{s}, \boldsymbol{w}, \boldsymbol{\pi}) &\propto p(r_{n,i}|\boldsymbol{r}_{-(n,i)}, \boldsymbol{\pi}) \\
&\quad \cdot p(w_{n,i}|\boldsymbol{r}, \boldsymbol{s}, \boldsymbol{w}_{-(n,i)}, \boldsymbol{z}, \boldsymbol{\beta}), \qquad (2) \\
p(z_{n,i}|\boldsymbol{r}, \boldsymbol{w}, \boldsymbol{z}_{-(n,i)}, \boldsymbol{\alpha}, \boldsymbol{\beta}) &\propto p(z_{n,i}|\boldsymbol{r}, \boldsymbol{z}_{-(n,i)}, \boldsymbol{\alpha}) \\
&\quad \cdot p(w_{n,i}|\boldsymbol{r}, \boldsymbol{w}_{-(n,i)}, \boldsymbol{z}, \boldsymbol{\beta}),
\end{aligned}$$

where $\boldsymbol{\pi}$ is a symmetric Dirichlet prior on session-wise word source distribution $\boldsymbol{\tau}_m$'s, and other symbols are defined above. All these probabilities are Dirichlet-multinomial distributions and therefore can be computed efficiently.

3.3 TM-HMMSS

The *TM-HMMSS* (Figure 2(c)) model modifies *TM-HMMS* to re-sample the distribution over word sources $\boldsymbol{\tau}$ *at every utterance*, instead of once at the beginning of each session. This modification allows the fraction of words drawn from the session-wide topics to vary over the course of the dialogue. This is attractive in task-oriented dialogue, where some sections of the dialogue always follow a similar script, regardless of session topic—for example, the opening, closing, or asking the user if they will take a survey. To support these patterns, *TM-HMMSS* conditions the source generator distribution on the current state.

The generative story of *TM-HMMSS* is very similar to *TM-HMMS*, except the distribution over word sources $\boldsymbol{\tau}$'s are sampled at every state. A dialogue is generated as following:

1: For each session, draw a topic distribution $\boldsymbol{\theta}$.
2: For each utterance n in the conversation, draw a state s_n based on previous state s_{n-1}, and subsequently retrieve the state-specific distribution over word sources $\boldsymbol{\tau}_{s_n}$.
3: For each word in utterance n, first sample a word source r according to $\boldsymbol{\tau}_{s_n}$, and then depending on r, generate a word w either from the session-wide topic distribution $\boldsymbol{\theta}$ or the language model specified by the state s_n.

As in *TM-HMMS*, we assume multinomial distributions $\boldsymbol{\theta}$'s and $\boldsymbol{\phi}$'s are drawn from Dirichlet priors; and $\boldsymbol{\tau}$'s are governed by Beta distributions.

The inference for *TM-HMMSS* is exactly same as the inference for *TM-HMMS*, except the posterior distributions over word source $r_{n,i}$ is now

$$\begin{aligned}
p(r_{n,i}|\boldsymbol{r}_{-(n,i)}, \boldsymbol{s}, \boldsymbol{w}, \boldsymbol{\pi}) &\propto p(r_{n,i}|\boldsymbol{r}_{-(n,i)}, s_n, \boldsymbol{\pi}) \\
&\quad \cdot p(w_{n,i}|\boldsymbol{r}, \boldsymbol{s}, \boldsymbol{w}_{-(n,i)}, \boldsymbol{z}, \boldsymbol{\beta}), \qquad (3)
\end{aligned}$$

where the first term is integrated over all sessions and conditioned on the state assignment.

3.4 Supporting Multiple Parties

Since our primary focus is task-oriented dialogues between two parties, we assume every word source is associated with *two* sets of *LM*s—one for system/agent and another for user/client. This configuration is similar to *PolyLDA* (Mimno et al., 2009) or *LinkLDA* (Yano et al., 2009), such that utterances from different parties are treated as different languages or blog-post and comments pairs. In this work, we implement all models under this setting, but omit details in plate diagrams for the sake of simplicity.

In settings where the agent and client always alternate, each state emits both text before transitioning to the next state. This is the case in the *BusTime* dataset, where the spoken dialogue system enforces strict turn-taking. In settings where agents or client may produce more than one utterance in a row, each state emits either agent text *or* client text, then transitions to the next state. This is the case in the *TechSupport* corpus, where either conversant may send a message at any time.

3.5 Likelihood Estimation

To evaluate performance across different models, we compute the likelihood on held-out test set. For *TM-HMM* model, there are no local dependencies, and we therefore compute the marginal likelihood using the forward algorithm. However, for *TM-HMMS* and *TM-HMMSS* models, the latent topic distribution $\boldsymbol{\theta}$ creates local dependencies, rendering computation of marginal likeli-

hoods intractable. Hence, we use a Chib-style estimator (Wallach et al., 2009). Although it is computationally more expensive, it gives less biased approximation of marginal likelihood, even for finite samples. This ensures likelihood measurements are comparable across models.

4 Experiments

In this section, we examine the effectiveness of our models. We first evaluate our models qualitatively by exploring the inferred state diagram. We then perform quantitative analysis with log likelihood measurements and an ordering task on a held-out test set. We train all models with 80% of the entire dataset and use the rest for testing. We run the Gibbs samplers for 1000 iterations and update all hyper-parameters using slice sampling (Neal, 2003; Wallach, 2008) every 10 iterations. The training likelihood suggest all models converge within $500-800$ iterations. For all Chib-style estimators, we collect 100 samples along the Markov chain to approximate the marginal likelihood.

4.1 Qualitative Evaluation

Figure 3 shows the state diagram for *BusTime* corpus inferred by *TM-HMM* without any supervision.[5] Every dialogue is opened by asking the user to say a bus route, or to say "I'm not sure." It then transits to a state about location, e.g., origin and destination. Both these two states may continue to a confirmation step immediately after. After verifying all the necessary information, the system asks if the user wants "the next few buses".[6] Otherwise, the system follows up with the user on the particular date and time information. After system reads out bus times, the user has options to "repeat" or ask for subsequent schedules.

In addition, we also include the human-annotated dialogue flow in Figure 4 for reference (Williams, 2012). It only illustrates the most common design of system actions, without showing edge cases. Comparing these two figures, the dialogue flow inferred by our model along the most probable path (highlighted in bold red in Figure 3) is consistent with underlying design. Furthermore, our models are able to capture edge cases—omitted for space—through a more general and probabilistic fashion. In summary, our models yield a very similar flowchart to the underlying design in a completely unsupervised way.[7]

Figure 5 shows part of the flowchart for the *TechSupport* corpus, generated by the *TM-HMMSS* model.[8] A conversation usually starts with a welcome message from a customer support agent. Next, clients sometimes report a problem; otherwise, the agent gathers the client's identity. After these preliminaries, the agent usually checks the system version or platform settings. Then, information about the problem is exchanged, and a cycle ensues where agents propose solutions, and clients attempt them, reporting results. Usually, a conversation loops among these states until either the problem is resolved (as the case shown in the figure) or the client is left with a reference number for future follow-up (not shown due to space limit). Although technical support is task-oriented, the scope of possible issues is vast and not prescribed. The table in Figure 5 lists the top ranked words of selected topics—the categories clients often report problems in. It illustrates that, qualitatively, *TM-HMMSS* discovers both problem categories and conversation structures on our data.

As one of the baseline model, we also include a part of flowchart generated by *LM-HMM* model with similar settings of $T = 20$ states. Illustrated by the highlighted states in 6, *LM-HMM* model conflates interactions that commonly occur at the beginning and end of a dialogue—i.e., "acknowledge agent" and "resolve problem", since their underlying language models are likely to produce similar probability distributions over words. By incorporating topic information, our proposed models (e.g., *TM-HMMSS* in Figure 5) are able to enforce the state transitions towards more frequent flow patterns, which further helps to overcome the weakness of language model.

4.2 Quantitative Evaluation

In this section, we evaluate our models using log likelihood and an ordering task on a held-out test set. Both evaluation metrics measure the predictive power of a conversation model.

[5] Recall in *BusTime*, state transitions occur after each *pair* of system/user utterances, so we display them synchronously.

[6] The system was designed this way because most users say "yes" to this question, obviating the date and time.

[7] We considered various ways of making a quantitative evaluation of the inferred state diagram, and proved difficult. Rather than attempt to justify a particular sub-division of each "design states", we instead give several straightforward quantitative evaluations in the next section.

[8] Recall in this corpus, state transitions occur after emitting each agent *or* client utterances, which does *not* necessarily alternate in a dialogue, so we display client request and agent response separately.

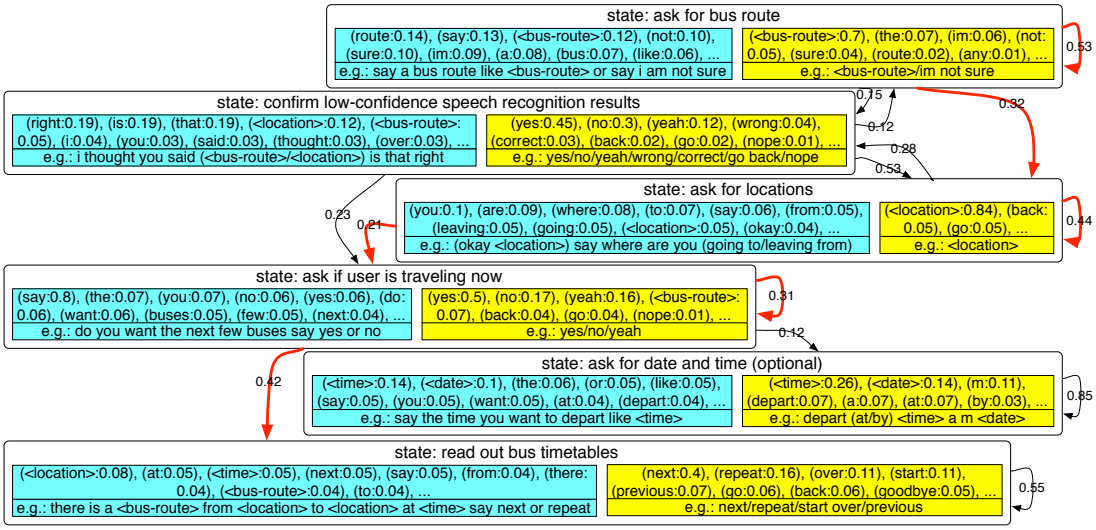


Figure 3: **(Upper)** Part of the flowchart inferred on *BusTime*, by *TM-HMM* model with $K = 10$ topics and $T = 10$ states. The most probable path is highlighted, which is consistent with the underlying design (Figure 4). Cyan blocks are system actions and yellow blocks are user responses. In every block, the upper cell shows the top ranked words marginalized over all topics and the lower cell shows some examples of that state. Transition probability cut-off is 0.1. States are labelled manually.

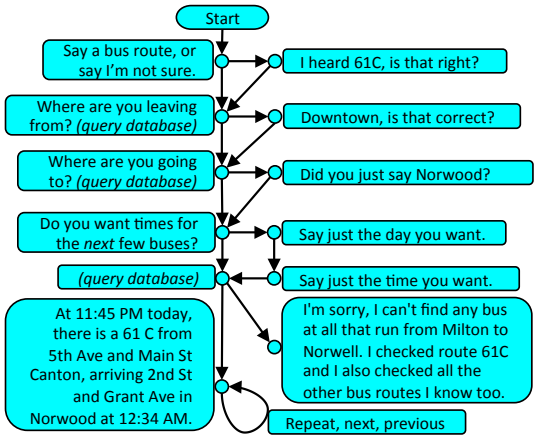


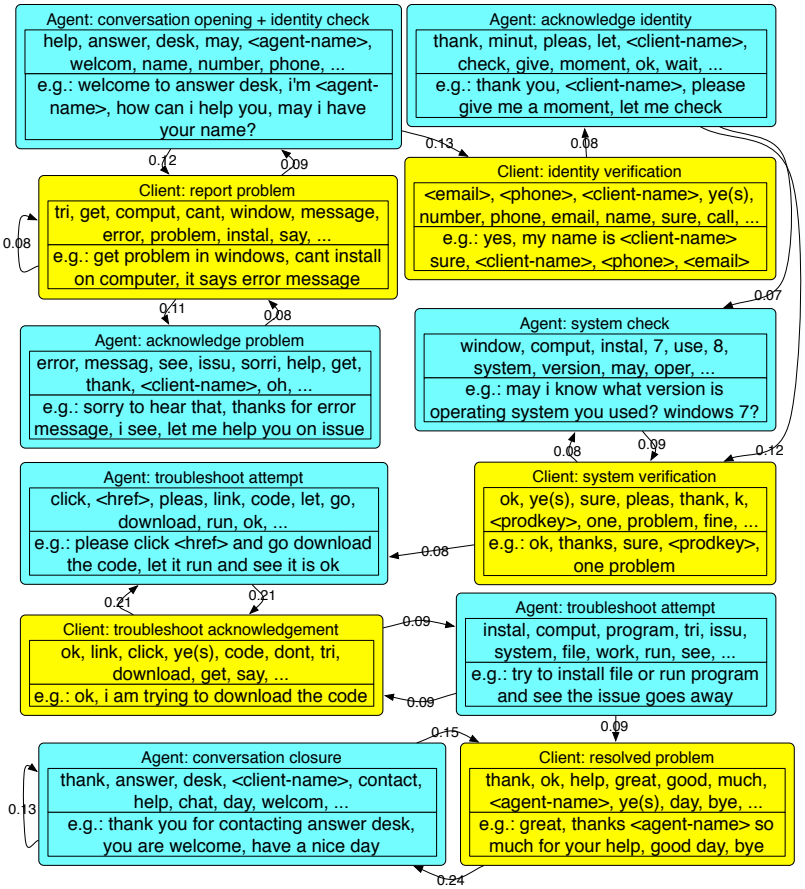
Figure 4: **(Left)** Hand-crafted reference flowchart for *BusTime* (Williams, 2012). Only the most common dialogue flows are displayed. System prompts shown are example paraphrases. Edge cases are not included.

Log Likelihood The likelihood metric measures the probability of generating the test set under a specified model. As shown in Figure 7, our models yield as good or better likelihood than *LM-HMM* and *LM-HMMS* models on both datasets under all settings. For our proposed models, *TM-HMMS* and *TM-HMMSS* perform better than *TM-HMM* on *TechSupport*, but not necessarily on *BusTime*. In addition, we notice that the marginal benefit of *TM-HMMSS* over *TM-HMM* is greater on *TechSupport* dataset, where each dialogue focuses on one of many possible tasks. This coincides with our belief that topics are more conversation dependent and shared across the entire corpus in customer support data—i.e., different clients in different sessions might ask about similar issues.

Ordering Test Ritter et al. (2010) proposes an evaluation based on rank correlation coefficient, which measures the degree of similarity between any two orderings over sequential data. They use Kendall's τ as evaluation metric, which is based on the agreement between pairwise orderings of two sequences (Kendall, 1938). It ranges from -1 to $+1$, where $+1$ indicates an identical ordering and -1 indicates a reverse ordering. The idea is to generate all permutations of the utterances in a dialogue (including true ordering), and compute the log likelihood for each under the model. Then, Kendall's τ is computed between the most probable permutation and true ordering. The result is the average of τ values for all dialogues in test corpus.

Ritter et al. (2010) limits their dataset by choosing Twitter dialogues containing 3 to 6 posts (utterances), making it tractable to enumerate all permutations. However, our datasets are much larger, and enumerating all possible permutations of dialogues with more than 20 or 30 utterances is infeasible. Instead, we incrementally build up the permutation set by adding one random permutation at a time, and taking the most probable permutation after each addition. If this process were continued (intractably!) until all permutations are enumerated, the true value of Kendall's τ test would be reached. In practice, the value appears to plateau after a few dozen measurements.

We present our results in Figure 8. Our models consistently perform as good or better than



Topic	Top Ranked Words
purchase	microsoft, store, purchas, able, get, sir, order, site, mr, contact, mac, ...
browser	internet, explor, browser, ie, open, websit, googl, download, click, chrome, ...
backup	file, restor, system, comput, back, folder, creat, option, dont, delet, ...
boot	comput, boot, mode, option, disc, safe, recoveri, repair, back, clean, cd, disk, ...
update	updat, window, servic, instal, pack, run, comput, download, check, restart, inform, system, error, fix, ...
network	connect, internet, printer, comput, network, pc, print, access, wireless, hp, cable, adapt, router, speed, ...
anti-virus	viru, scan, comput, remov, secur, run, system, anti, essenti, infect, defend, softwar, program, protect, antiviru, malwar, ...
hardware	driver, devic, drive, dvd, cd, hardwar, issu, model, laptop, plug, software, usb, ...
windows	window, upgrad, 8, download, 7, instal, bit, vista, pro, system, ...
office	offic, 2010, word, microsoft, home, excel, version, 2007, student, document, trial, 2013, ...
outlook	outlook, account, email, mail, microsoft, com, live, password, profil, contact, creat, server, access, ...
license	key, product, activ, purchas, licens, valid, verifi, id, disc, pro, grenuin, ...
facility	window, 8, comput, instal, manufactur, system, oem, 7, pc, hp, ...

Figure 5: Part of flowchart (left) and topic table (right) on *TechSupport* dataset, generated by *TM-HMMSS* model under settings of $K = 20$ topics and $T = 20$ states. The topic table lists top ranked words in issues discussed in the chats. Cyan blocks are system actions and yellow blocks are user responses. In every block, the upper cell shows top ranked words, and the lower cell shows example string patterns of that state. Transition probability cut-off is 0.05. States and topics are labelled manually.

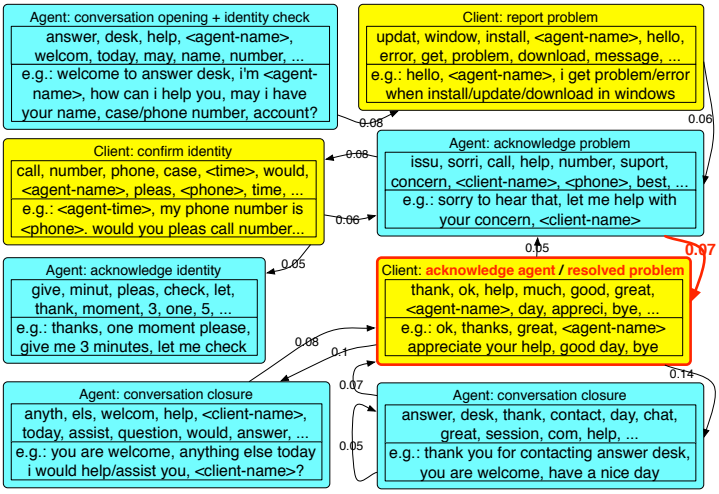


Figure 6: Part of flowchart on *TechSupport* dataset, generated by *LM-HMM* model with $T = 20$ states. Cyan blocks are system actions and yellow blocks are user responses. In every block, the upper cell shows the top ranked words, and the lower cell shows example word sequences or string patterns of that state. Transition probability cut-off is 0.05. States are labelled manually. A poorly-inferred state is highlighted, which seems to conflate the "acknowledge agent" and "resolve problem" states, and *TM-HMMSS* model has properly disentangled (Figure 5).

the baseline models. For *BusTime* data, all models perform relatively well except *LM-HMM* which only indicates weak correlations. *TM-HMM* out-performs all other models under all settings. This is also true for *TechSupport* dataset. *LM-HMMS*, *TM-HMMS* and *TM-HMMSS* models perform considerably well on *BusTime*, but not on *TechSupport* data. These three models allow words to be generated from additional sources other than states. Although this improves log likelihood, it is possible these models encode less information about the state sequences, at least in the more diffuse *TechSupport* data. In summary, under both quantitative evaluation measures, our models advance state-of-the-art, however *which* of our models is best depends on the application.

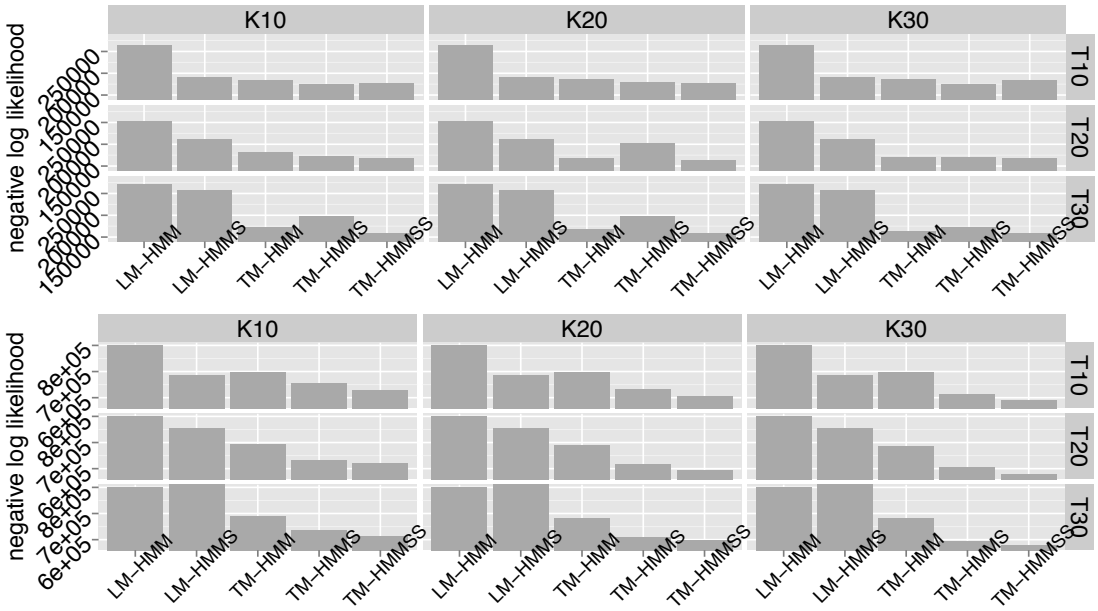


Figure 7: Negative log likelihood on *BusTime* (upper) and *TechSupport* (lower) datasets (smaller is better) under different settings of topics K and states T.

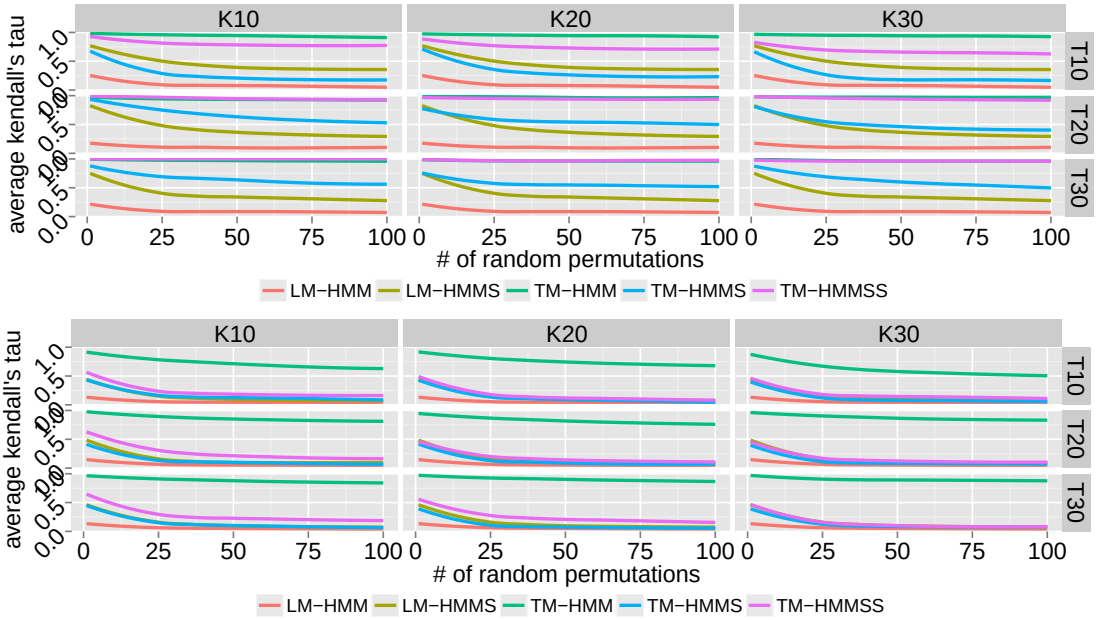


Figure 8: Average Kendall's τ measure on *BusTime* (upper) and *TechSupport* (lower) datasets (larger is better) against number of random permutations, under various settings of topics K and states T.

5 Conclusion and Future Work

We have presented three new unsupervised models to discover latent structures in task-oriented dialogues. We evaluated on two very different corpora—logs from spoken, human-computer dialogues about bus time, and logs of textual, human-human dialogues about technical support. We have shown our models yield superior performance both qualitatively and quantitatively.

One possible avenue for future work is scalability. Parallelization (Asuncion et al., 2012) or online learning (Doucet et al., 2001) could significantly speed up inference. In addition to MCMC, another class of inference method is variational Bayesian analysis (Blei et al., 2003; Beal, 2003), which is inherently easier to distribute (Zhai et al., 2012) and online update (Hoffman et al., 2010).

Acknowledgments

We would like to thank anonymous reviewers and Jordan Boyd-Graber for their valuable comments. We are also grateful to Alan Ritter and Bill Dolan for their helpful discussions; and Kai (Anthony) Lui for providing *TechSupport* dataset.

References

Arthur Asuncion, Padhraic Smyth, Max Welling, David Newman, Ian Porteous, and Scott Triglia, 2012. *Distributed Gibbs sampling for latent variable models*.

Satanjeev Banerjee and Alexander I Rudnicky. 2006. A texttiling based approach to topic boundary detection in meetings. In *INTERSPEECH*.

Srinivas Bangalore, Giuseppe Di Fabbrizio, and Amanda Stent. 2006. Learning the structure of task-driven human-human dialogs. In *ACL*, Stroudsburg, PA, USA.

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *NAACL*, pages 113–120.

Matthew J. Beal. 2003. *Variational Algorithms for Approximate Bayesian Inference*. Ph.D. thesis.

Alan W Black, Susanne Burger, Alistair Conkie, Helen Hastie, Simon Keizer, Nicolas Merigaud, Gabriel Parent, Gabriel Schubiner, Blaise Thomson, D. Williams, Kai Yu, Steve Young, and Maxine Eskenazi. 2010. Spoken dialog challenge 2010: Comparison of live and control test results. In *SIGDIAL*.

David M. Blei, Andrew Ng, and Michael Jordan. 2003. Latent Dirichlet allocation. *JMLR*.

Ananlada Chotimongkol. 2008. *Learning the Structure of Task-oriented Conversations from the Corpus of In-domain Dialogs*. Ph.D. thesis.

Nigel Crook, Ramn Granell, and Stephen G. Pulman. 2009. Unsupervised classification of dialogue acts using a dirichlet process mixture model. In *SIGDIAL*.

Hal Daumé III and Daniel Marcu. 2006. Bayesian query-focused summarization. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 305–312, Morristown, NJ, USA. Association for Computational Linguistics.

David DeVault, Kallirroi Georgila, Ron Artstein, Fabrizio Morbini, David Traum, Stefan Scherer, Albert Rizzo, and Louis-Philippe Morency. 2013. Verbal indicators of psychological distress in interactive dialogue with a virtual human. In *SIGDIAL*.

Arnaud Doucet, Nando De Freitas, and Neil Gordon, editors. 2001. *Sequential Monte Carlo methods in practice*. Springer Texts in Statistics.

Michel Galley, Kathleen McKeown, Eric Fosler-Lussier, and Hongyan Jing. 2003. Discourse segmentation of multi-party conversation. In *ACL*.

Sharon Goldwater and Thomas L. Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *ACL*.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *PNAS*, 101(Suppl 1):5228–5235.

James Henderson and Oliver Lemon. 2008. Mixture model POMDPs for efficient handling of uncertainty in dialogue management. In *ACL*.

Matthew Hoffman, David M. Blei, and Francis Bach. 2010. Online learning for latent Dirichlet allocation. In *NIPS*.

Pei-yun Hsueh, Johanna D. Moore, and Steve Renals. 2006. Automatic segmentation of multiparty dialogue. In *EACL*.

Lluís F. Hurtado, Joaquin Planells, Encarna Segarra, Emilio Sanchis, and David Griol. 2010. A stochastic finite-state transducer approach to spoken dialog management. In *INTERSPEECH*.

Dan Jurafsky, Elizabeth Shriberg, and Debra Biasca. 1997. Switchboard SWBD-DAMSL shallow-discourse-function annotation coders manual. *Institute of Cognitive Science Technical Report*, pages 97–02.

Maurice G. Kendall. 1938. A new measure of rank correlation. Biometrika Trust.

Staffan Larsson and David R. Traum. 2000. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering*, 5(3/4):323–340.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. 2000. A stochastic model of human-machine interaction for learning dialogue strategies. *IEEE Trans on Speech and Audio Processing*, 8(1):11–23.

Jingjing Liu, Stephanie Seneff, and Victor Zue. 2010. Dialogue-oriented review summary generation for spoken dialogue recommendation systems. In *NAACL*.

David Mimno, Hanna Wallach, Jason Naradowsky, David Smith, and Andrew McCallum. 2009. Polylingual topic models. In *EMNLP*.

Gabriel Murray, Steve Renals, and Jean Carletta. 2005. Extractive summarization of meeting recordings. In *European Conference on Speech Communication and Technology*.

Radford M. Neal. 2000. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265.

Radford M. Neal. 2003. Slice sampling. *Annals of Statistics*, 31:705–767.

Matthew Purver, Konrad Körding, Thomas L. Griffiths, and Joshua Tenenbaum. 2006. Unsupervised topic modelling for multi-party spoken discourse. In *ACL*.

Verena Rieser and Oliver Lemon. 2010. Natural language generation as planning under uncertainty for spoken dialogue systems. In *EMNLP*.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of twitter conversations. In *NAACL*.

Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. 2002. Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artificial Intelligence Research*.

Andreas Stolcke, Noah Coccaro, Rebecca Bates, Paul Taylor, Carol Van Ess-Dykema, Klaus Ries, Elizabeth Shriberg, Daniel Jurafsky, Rachel Martin, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, September.

David R Traum and Staffan Larsson. 2003. The information state approach to dialogue management. In *Current and new directions in discourse and dialogue*, pages 325–353.

Hanna M. Wallach, Iain Murray, Ruslan Salakhutdinov, and David Mimno. 2009. Evaluation methods for topic models. In *ICML*.

Hanna M. Wallach. 2008. *Structured Topic Models for Language*. Ph.D. thesis, University of Cambridge.

Jason D. Williams and Suhrid Balakrishnan. 2009. Estimating probability of correctness for ASR N-best lists. In *SIGDIAL*.

Jason D. Williams. 2012. Challenges and opportunities for state tracking in statistical spoken dialog systems: Results from two public deployments. *Journal of Selected Topics in Signal Processing*.

Tae Yano, William W. Cohen, and Noah A. Smith. 2009. Predicting response to political blog posts with topic models. In *NAACL*, pages 477–485, Stroudsburg, PA, USA. ACL.

Steve Young. 2006. Using POMDPs for dialog management. In *Proceedings of the 1st IEEE/ACL Workshop on Spoken Language Technologies (SLT06)*.

Ke Zhai, Jordan Boyd-Graber, Nima Asadi, and Mohamad Alkhouja. 2012. Mr. LDA: A flexible large scale topic modeling package using variational inference in mapreduce. In *WWW*.

Learning Structured Perceptrons for Coreference Resolution with Latent Antecedents and Non-local Features

Anders Björkelund and **Jonas Kuhn**
Institute for Natural Language Processing
University of Stuttgart
{anders,jonas}@ims.uni-stuttgart.de

Abstract

We investigate different ways of learning structured perceptron models for coreference resolution when using non-local features and beam search. Our experimental results indicate that standard techniques such as early updates or Learning as Search Optimization (LaSO) perform worse than a greedy baseline that only uses local features. By modifying LaSO to delay updates until the end of each instance we obtain significant improvements over the baseline. Our model obtains the best results to date on recent shared task data for Arabic, Chinese, and English.

1 Introduction

This paper studies and extends previous work using the structured perceptron (Collins, 2002) for complex NLP tasks. We show that for the task of coreference resolution the straightforward combination of beam search and early update (Collins and Roark, 2004) falls short of more limited feature sets that allow for exact search. This contrasts with previous work on, e.g., syntactic parsing (Collins and Roark, 2004; Huang, 2008; Zhang and Clark, 2008) and linearization (Bohnet et al., 2011), and even simpler structured prediction problems, where early updates are not even necessary, such as part-of-speech tagging (Collins, 2002) and named entity recognition (Ratinov and Roth, 2009).

The main reason why early updates underperform in our setting is that the task is too difficult and that the learning algorithm is not able to profit from all training data. Put another way, early updates happen too early, and the learning algorithm rarely reaches the end of the instances as it halts, updates, and moves on to the next instance.

An alternative would be to continue decoding the same instance after the early updates, which is equivalent to *Learning as Search Optimization* (LaSO; Daumé III and Marcu (2005b)). The learning task we are tackling is however further complicated since the target structure is under-determined by the gold standard annotation. Coreferent mentions in a document are usually annotated as sets of mentions, where all mentions in a set are coreferent. We adopt the recently popularized approach of inducing a latent structure within these sets (Fernandes et al., 2012; Chang et al., 2013; Durrett and Klein, 2013). This approach provides a powerful boost to the performance of coreference resolvers, but we find that it does not combine well with the LaSO learning strategy. We therefore propose a modification to LaSO, which delays updates until after each instance. The combination of this modification with non-local features leads to further improvements in the clustering accuracy, as we show in evaluation results on all languages from the CoNLL 2012 Shared Task – Arabic, Chinese, and English. We obtain the best results to date on these data sets.[1]

2 Background

Coreference resolution is the task of grouping referring expressions (or *mentions*) in a text into disjoint *clusters* such that all mentions in a cluster refer to the same entity. An example is given in Figure 1 below, where mentions from two clusters are marked with brackets:

> [Drug Emporium Inc.]$_{a_1}$ said [Gary Wilber]$_{b_1}$ was named CEO of [this drugstore chain]$_{a_2}$. [He]$_{b_2}$ succeeds his father, Philip T. Wilber, who founded [the company]$_{a_3}$ and remains chairman. Robert E. Lyons III, who headed the [company]$_{a_4}$'s Philadelphia region, was appointed president and chief operating officer, succeeding [Gary Wilber]$_{b_3}$.

Figure 1: An excerpt of a document with the mentions from two clusters marked.

[1] Our system is available at http://www.ims.uni-stuttgart.de/~anders/coref.html

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 47–57,
Baltimore, Maryland, USA, June 23-25 2014.

In recent years much work on coreference resolution has been devoted to increasing the expressivity of the classical *mention-pair model*, in which each coreference classification decision is limited to information about two mentions that make up a pair. This shortcoming has been addressed by *entity-mention models*, which relate a candidate mention to the full cluster of mentions predicted to be coreferent so far (for more discussion on the model types, see, e.g., (Ng, 2010)).

Nevertheless, the two best systems in the latest CoNLL Shared Task on coreference resolution (Pradhan et al., 2012) were both variants of the mention-pair model. While the second best system (Björkelund and Farkas, 2012) followed the widely used baseline of Soon et al. (2001), the winning system (Fernandes et al., 2012) proposed the use of a **tree representation**.

The tree-based model of Fernandes et al. (2012) construes the representation of coreference clusters as a rooted tree. Figure 2 displays an example tree over the clusters from Figure 1. Every mention corresponds to a node in the tree, and arcs between mentions indicate that they are coreferent. The tree additionally has a dummy root node. Every subtree under the root node corresponds to a cluster of coreferent mentions.

Since coreference training data is typically not annotated with trees, Fernandes et al. (2012) proposed the use of **latent** trees that are induced during the training phase of a coreference resolver. The latent tree provides more meaningful antecedents for training.[2] For instance, the popular pair-wise instance creation method suggested by Soon et al. (2001) assumes non-branching trees, where the antecedent of every mention is its linear predecessor (i.e., *he*$_{b_2}$ is the antecedent of *Gary Wilber*$_{b_3}$). Comparing the two alternative antecedents of *Gary Wilber*$_{b_3}$, the tree in Figure 2 provides a more reliable basis for training a coreference resolver, as the two mentions of *Gary Wilber* are both proper names and have an exact string match.

3 Representation and Learning

Let $M = \{m_0, m_1, ..., m_n\}$ denote the set of mentions in a document, including the artificial root mention (denoted by m_0). We assume that the mentions are ordered ascendingly with respect to the linear order of the document, where the document root precedes all other mentions.[3] For each mention m_j, let A_j denote the set of potential antecedents. That is, the set of all mentions that precede m_j according to the linear order including the root node, or, $A_j = \{m_i \mid i < j\}$. Finally, let $\mathcal{A}$ denote the set of all antecedent sets $\{A_0, A_1, ..., A_n\}$.

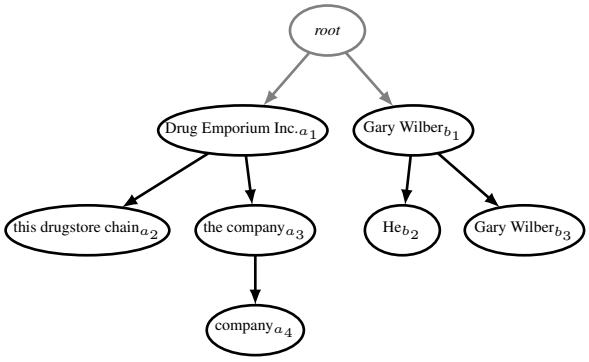


Figure 2: A tree representation of Figure 1.

In the tree model, each mention corresponds to a node, and an antecedent-anaphor pair $\langle a_i, m_i \rangle$, where $a_i \in A_i$, corresponds to a directed edge (or *arc*) pointing from antecedent to anaphor.

The score of an arc $\langle a_i, m_i \rangle$ is defined as the scalar product between a weight vector w and a feature vector $\Phi(\langle a_i, m_i \rangle)$, where Φ is a feature extraction function over an arc (thus extracting features from the antecedent and the anaphor). The score of a coreference tree $y = \{\langle a_1, m_1 \rangle, \langle a_2, m_2 \rangle, ..., \langle a_n, m_n \rangle\}$ is defined as the sum of the scores of all the mention pairs:

$$score(\langle a_i, m_i \rangle) = w \cdot \Phi(\langle a_i, m_i \rangle) \quad (1)$$
$$score(y) = \sum_{\langle a_i, m_i \rangle \in y} score(\langle a_i, m_i \rangle)$$

The objective is to find the output $\hat{y}$ that maximizes the scoring function:

$$\hat{y} = \arg\max_{y \in \mathcal{Y}(\mathcal{A})} score(y) \quad (2)$$

where $\mathcal{Y}(\mathcal{A})$ denotes the set of possible trees given the antecedent sets $\mathcal{A}$. By treating the mentions as nodes in a directed graph and assigning scores to the arcs according to (1), Fernandes et al. (2012) solved the search problem using the Chu-Liu-Edmonds (CLE) algorithm (Chu and Liu, 1965;

[2] We follow standard practice and overload the terms anaphor and antecedent to be any type of mention, i.e., names as well as pronouns. An antecedent is simply the mention to the left of the anaphor.

[3] We impose a total order on mentions. In case of nested mentions, the mention that begins first is assumed to precede the embedded one. If two mentions begin at the same token, the longer one is taken to precede the shorter one.

Edmonds, 1967), which is a maximum spanning tree algorithm that finds the optimal tree over a connected directed graph. CLE, however, has the drawback that the scores of the arcs must remain fixed and can not change depending on other arcs and it is not clear how to include non-local features in a CLE decoder.

3.1 Online learning

We find the weight vector w by online learning using a variant of the structured perceptron (Collins, 2002). Specifically, we use the passive-aggressive (PA) algorithm (Crammer et al., 2006), since we found that this performed slightly better in preliminary experiments.[4]

The structured perceptron iterates over training instances $\langle x_i, y_i \rangle$, where x_i are inputs and y_i are outputs. For each instance it uses the current weight vector w to make a prediction $\hat{y}_i$ given the input x_i. If the prediction is incorrect, the weight vector is updated in favor of the correct structure. Otherwise the weight vector is left untouched. In our setting inputs x_i correspond to documents and outputs y_i are trees over mentions in a document. The training data is, however, not annotated with trees, but only with clusters of mentions. That is, the y_i's are not defined a priori.

3.2 Latent antecedents

In order to have a tree structure to update against, we use the current weight vector and apply the decoder to a constrained antecedent set and obtain a **latent tree** over the mentions in a document, where each mention is assigned a single correct antecedent (Fernandes et al., 2012). We constrain the antecedent sets such that only trees that correspond to the correct clustering can be built. Specifically, let $\tilde{A}_j$ denote the set of correct antecedents for a mention m_j, or

$$\tilde{A}_j = \begin{cases} \{m_0\} \text{ if } m_j \text{ has no correct antecedent} \\ \{a_i \mid \textsc{Coref}(a_i, m_j), a_i \in A_j\} \text{ otherwise} \end{cases}$$

that is, if mention m_j is non-referential or the first mention of its cluster, $\tilde{A}_j$ contains only the document root. Otherwise it is the set of all mentions to the left that belong to the same cluster as m_j. Analogously to $\mathcal{A}$, let $\tilde{\mathcal{A}}$ denote the set of constrained antecedent sets. The latent tree $\tilde{y}$ needed for updates is then defined to be the optimal tree over $\mathcal{Y}(\tilde{\mathcal{A}})$, subject to the current weight vector:

$$\tilde{y} = \arg\max_{y \in \mathcal{Y}(\tilde{\mathcal{A}})} score(y)$$

The intuition behind the latent tree is that during online learning, the weight vector will start favoring latent trees that are easier to learn (such as the one in Figure 2).

Algorithm 1 PA algorithm with latent trees

Input: Training data D, number of iterations T
Output: Weight vector w

1: $w = \overrightarrow{0}$
2: **for** $t \in 1..T$ **do**
3: **for** $\langle M_i, \mathcal{A}_i, \tilde{\mathcal{A}}_i \rangle \in D$ **do**
4: $\hat{y}_i = \arg\max_{\mathcal{Y}(\mathcal{A})} score(y)$ ▷ Predict
5: **if** $\neg$ CORRECT$(\hat{y}_i)$ **then**
6: $\tilde{y}_i = \arg\max_{\mathcal{Y}(\tilde{\mathcal{A}})} score(y)$ ▷ Latent tree
7: $\Delta = \Phi(\hat{y}_i) - \Phi(\tilde{y}_i)$
8: $\tau = \frac{\Delta \cdot w + \text{Loss}(\hat{y}_i)}{\|\Delta\|^2}$ ▷ PA weight
9: $w = w + \tau\Delta$ ▷ PA update
10: **return** w

Algorithm 1 shows pseudocode for the learning algorithm, which we will refer to as the **baseline learning algorithm**. Instead of looping over pairs $\langle x, y \rangle$ of documents and trees, it loops over triples $\langle M, \mathcal{A}, \tilde{\mathcal{A}} \rangle$ that comprise the set of mentions M and the two sets of antecedent candidates (line 3). Moreover, rather than checking that the tree is identical to the latent tree, it only requires the tree to correctly encode the gold clustering (line 5). The update that occurs in lines 7-9 is the passive-aggressive update. A loss function LOSS that quantifies the error in the prediction is used to compute a scalar τ that controls how much the weights are moved in each update. If τ is set to 1, the update reduces to the standard structured perceptron update. The loss function can be an arbitrarily complex function that returns a numerical value of how bad the prediction is. In the simplest case, Hamming loss can be used, i.e., for each incorrect arc add 1. We follow Fernandes et al. (2012) and penalize erroneous root attachments, i.e., mentions that erroneously get the root node as their antecedent, with a loss of 1.5. For all other arcs we use Hamming loss.

4 Incremental Search

We now show that the search problem in (2) can equivalently be solved by the more intuitive **best-first decoder** (Ng and Cardie, 2002), rather than using the CLE decoder. The best-first decoder

[4] We also implement the feature mapping function Φ as a *hash kernel* (Bohnet, 2010) and apply *averaging* (Collins, 2002), though for brevity we omit this from the pseudocode.

works incrementally by making a left-to-right pass over the mentions, selecting for each mention the highest scoring antecedent.

The key aspect that makes the best-first decoder equivalent to the CLE decoder is that all arcs point from left to right, both in this paper and in the work of Fernandes et al. (2012). We sketch a proof that this decoder also returns the highest scoring tree.

First, note that this algorithm indeed returns a tree. This can be shown by assuming the opposite, in which case the tree has to have a cycle. Then there must be a mention that has its antecedent to the right. Though this is not possible since all arcs point from left to right.

Second, this tree is the highest scoring tree. Again, assume the contrary, i.e., that there is a higher scoring tree in $\mathcal{Y}(\mathcal{A})$. This implies that for some mention there is a higher scoring antecedent than the one selected by the decoder. This contradicts the fact that the best-first decoder selects the highest scoring antecedent for each mention.[5]

5 Introducing Non-local Features

Since the best-first decoder makes a left-to-right pass, it is possible to extract features on the partial structure on the left. Such **non-local features** are able to capture information beyond that of a mention and its potential antecedent, e.g., the size of a partially built cluster, or features extracted from the antecedent of the antecedent.

When only local features are used, greedy search (either with CLE or the best-first decoder) suffices to find the highest scoring tree. That is, greedy search provides an exact solution to equation 2. Non-local features, however, render the exact search problem intractable. This is because with non-local features, locally suboptimal (i.e., non-greedy) antecedents for some mentions may lead to a higher total score over a whole document.

In order to keep some options around during search, we extend the best-first decoder with **beam search**. Beam search works incrementally by keeping an *agenda* of *state items*. At each step, all items on the agenda are expanded. The subset of size k (the *beam size*) of the highest scoring expansions are retained and put back into the agenda for the next step. The feature extraction function Φ is also extended such that it also receives the current state s as an argument: $\Phi(\langle m_i, m_j \rangle, s)$. The state encodes the previous decisions and enables Φ to extract features from the partial tree on the left.

We now outline three different ways of learning the weight vector w with non-local features.

5.1 Early updates

The beam search decoder can be plugged into the training algorithm, replacing the calls to $\arg\max$. Since state items leading to the best tree may be pruned from the agenda before the decoder reaches the end of the document, the introduction of non-local features may cause the decoder to return a non-optimal tree. This is problematic as it might cause updates although the correct tree has a higher score than the predicted one. It has previously been observed (Huang et al., 2012) that substantial gains can be made by applying an **early update** strategy (Collins and Roark, 2004): if the correct item is pruned before reaching the end of the document, then stop and update.

While beam search and early updates have been successfully applied to other NLP applications, our task differs in two important aspects: First, coreference resolution is a much more difficult task, which relies on more (world) knowledge than what is available in the training data. In other words, it is unlikely that we can devise a feature set that is informative enough to allow the weight vector to converge towards a solution that lets the learning algorithm see the entire documents during training, at least in the situation when no external knowledge sources are used.

Second, our gold structure is not known but is induced latently, and may vary from iteration to iteration. With non-local features this is troublesome since the best latent tree of a complete document may not necessarily coincide with the best *partial* tree at some intermediate mention m_j, $j < n$, i.e., a mention before the last in a document. We therefore also apply beam search to find the latent tree to have a partial gold structure for every mention in a document.

Algorithm 2 shows pseudocode for the beam search and early update training procedure. The algorithm maintains two parallel agendas, one for gold items and one for predicted items. At every mention, both agendas are expanded and thus cover the same set of mentions. Then the predicted agenda is checked to see if it contains any correct

[5] In case there are multiple maximum spanning trees, the best-first decoder will return one of them. This also holds for the CLE algorithm. With proper definitions, the proof can be constructed to show that both search algorithms return trees belonging to the *set* of maximum spanning trees over a graph.

Algorithm 2 Beam search and early update

Input: Data set D, epochs T, beam size k
Output: weight vector w

```
 1: w = 0⃗
 2: for t ∈ 1..T do
 3:     for ⟨M_i, A_i, Ã_i⟩ ∈ D do
 4:         Agenda_G = {}
 5:         Agenda_P = {}
 6:         for j ∈ 1..n do
 7:             Agenda_G = EXPAND(Agenda_G, Ã_j, m_j, k)
 8:             Agenda_P = EXPAND(Agenda_P, A_j, m_j, k)
 9:             if ¬ CONTAINSCORRECT(Agenda_P) then
10:                 ỹ = EXTRACTBEST(Agenda_G)
11:                 ŷ = EXTRACTBEST(Agenda_P)
12:                 update                          ▷ PA update
13:                 GOTO 3   ▷ Skip and move to next instance
14:         ŷ = EXTRACTBEST(Agenda_P)
15:         if ¬ CORRECT(ŷ) then
16:             ỹ = EXTRACTBEST(Agenda_G)
17:             update                              ▷ PA update
```

item. If there is no correct item in the predicted agenda, search is halted and an update is made against the best item from the gold agenda. The algorithm then moves on to the next document. If the end of a document is reached, the top scoring predicted item is checked for correctness. If it is not, an update is made against the best gold item.

A drawback of early updates is that the remainder of the document is skipped when an early update is applied, effectively discarding some training data.[6] An alternative strategy that makes better use of the training data is to apply the max-violation procedure suggested by Huang et al. (2012). However, since our gold trees change from iteration to iteration, and even inside of a single document, it is not entirely clear with respect to what gold tree the maximum violation should be computed. Initial experiments with max-violation updates indicated that they did not improve much over early updates, and also had a tendency to only consider a smaller portion of the training data.

5.2 LaSO

To make full use of the training data we implemented Learning as Search Optimization (**LaSO**; Daumé III and Marcu, 2005b). It is very similar to early updates, but differs in one crucial respect: When an early update is made, search is continued rather than aborted. Thus the learning algorithm always reaches the end of a document, avoiding the problem that early updates discard parts of the training data.

[6] In fact, after 50 iterations about 70% of the mentions in the training data are still being ignored due to early updates.

Correct items are computed the same way as with early updates, where an agenda of gold items is maintained in parallel. When search is resumed after an intermediate LaSO update, the prediction agenda is re-seeded with gold items (i.e., items that are all correct). This is necessary since the update influences what the partial gold structure looks like, and the gold agenda therefore needs to be recreated from the beginning of the document. Specifically, after each intermediate LaSO update, the gold agenda is expanded repeatedly from the beginning of the document to the point where the update was made, and is then copied over to seed the prediction agenda. In terms of pseudocode, this is accomplished by replacing lines 12 and 13 in Algorithm 2 with the following:

```
12:                 update                          ▷ PA update
13:                 Agenda_G = {}
14:                 for m_i ∈ {m_1, ..., m_j}   ▷ Recreate gold agenda
15:                     Agenda_G = EXPAND(Agenda_G, Ã_i, m_i, k)
16:                 Agenda_P = COPY(Agenda_G)
17:                 GOTO 6                          ▷ Continue
```

5.3 Delayed LaSO updates

When we applied LaSO, we noticed that it performed worse than the baseline learning algorithm when only using local features. We believe that the reason is that updates are made in the middle of documents which means that lexical forms of antecedents are "fresh in memory" of the weight vector. This results in fewer mistakes during training and leads to fewer updates. While this feedback makes it easier during training, such feedback is not available during test time, and the LaSO learning setting therefore mimics the testing setting to a lesser extent.

We also found that LaSO updates change the shape of the latent tree and that the average distance between mentions connected by an arc increased. This problem can also be attributed to how lexical items are fresh in memory. Such trees tend to deviate from the intuition that the latent trees are easier to learn. They also render distance-based features (which are standard practice and generally rather useful) less powerful, as distance in sentences or mentions becomes less of a reliable indicator for coreference.

To cope with this problem, we devised the **delayed LaSO** update, which differs from LaSO only in the respect that it postpones the actual updates until the end of a document. This is accomplished by summing the distance vectors Δ at every point where LaSO would make an update. At

Algorithm 3 Delayed LaSO update

Input: Data set D, iterations T, beam size k
Output: weight vector w

$w = \overrightarrow{0}$
for $t \in 1..T$ **do**
  for $\langle M_i, \mathcal{A}_i, \tilde{\mathcal{A}}_i \rangle \in D$ **do**
    $Agenda_G = \{\}$
    $Agenda_P = \{\}$
    $\Delta_{acc} = \overrightarrow{0}$
    $loss_{acc} = 0$
    for $j \in 1..n$ **do**
      $Agenda_G = \text{EXPAND}(Agenda_G, \tilde{A}_j, m_j, k)$
      $Agenda_P = \text{EXPAND}(Agenda_P, A_j, m_j, k)$
      if $\neg\,\text{CONTAINSCORRECT}(Agenda_P)$ **then**
        $\tilde{y} = \text{EXTRACTBEST}(Agenda_G)$
        $\hat{y} = \text{EXTRACTBEST}(Agenda_P)$
        $\Delta_{acc} = \Delta_{acc} + \Phi(\hat{y}) - \Phi(\tilde{y})$
        $loss_{acc} = loss_{acc} + \text{LOSS}(\hat{y})$
        $Agenda_P = Agenda_G$
    $\hat{y} = \text{EXTRACTBEST}(Agenda_P)$
    if $\neg\,\text{CORRECT}(\hat{y})$ **then**
      $\tilde{y} = \text{EXTRACTBEST}(Agenda_G)$
      $\Delta_{acc} = \Delta_{acc} + \Phi(\hat{y}) - \Phi(\tilde{y})$
      $loss_{acc} = loss_{acc} + \text{LOSS}(\hat{y})$
    if $\Delta_{acc} \neq \overrightarrow{0}$ **then**
      update w.r.t. Δ_{acc} and $loss_{acc}$

the end of a document, an update is made with respect to the sum of all Δ's. Similarly, a running sum of the partial loss is maintained within a document. Since the PA update only depends on the distance vector Δ and the loss, it can be applied with respect to these sums at the end of the document. When only local features are used, this update is equivalent to the updates in the baseline learning algorithm. This follows because greedy search finds the optimal tree when only local features are used. Similarly, using only local features, the beam-based best-first decoder will also return the optimal tree. Algorithm 3 shows the pseudocode for the delayed LaSO learning algorithm.

6 Features

In this section we briefly outline the type of features we use. The feature sets are customized for each language. As a baseline we use the features from Björkelund and Farkas (2012), who ranked second in the 2012 CoNLL shared task and is publicly available. The exact definitions and feature sets that we use are available as part of the download package of our system.

6.1 Local features

Basic features that can be extracted on one or both mentions in a pair include (among others): **Mention type**, which is either root, pronoun, name, or common; **Distance** features, e.g., the distance in sentences or mentions; **Rule-based** features, e.g., StringMatch or SubStringMatch; **Syntax-based** features, e.g., category labels or paths in the syntax tree; **Lexical** features, e.g., the head word of a mention or the last word of a mention.

In order to have a strong local baseline, we applied greedy forward/backward feature selection on the training data using a large set of local feature templates. Specifically, the training set of each language was split into two parts where 75% was used for training, and 25% for testing. Feature templates were incrementally added or removed in order to optimize the mean of MUC, B^3, and $CEAF_e$ (i.e., the CoNLL average).

6.2 Non-local Features

We experimented with non-local features drawn from previous work on entity-mention models (Luo et al., 2004; Rahman and Ng, 2009), however they did not improve performance in preliminary experiments. The one exception is the size of a cluster (Culotta et al., 2007). Additional features we use are

Shape encodes the linear "shape" of a cluster in terms of mention type. For instance, the clusters representing *Gary Wilber* and *Drug Emporium Inc.* from the example in Figure 1, would be represented as RNPN and RNCCC, respectively. Where R, N, P, and C denote the root node, names, pronouns, and common noun phrases, respectively.

Local syntactic context is inspired by the Entity Grid (Barzilay and Lapata, 2008), where the basic assumption is that references to an entity follow particular syntactic patterns. For instance, an entity may be introduced as an object in one sentence, whereas in subsequent sentences it is referred to in subject position. Grammatical functions are approximated by the path in the syntax tree from a mention to its closest S node. The partial paths of a mention and its linear predecessor, given the cluster of the current antecedent, informs the model about the local syntactic context.

Cluster start distance denotes the distance in mentions from the beginning of the document where the cluster of the antecedent in consideration begins.

Additionally, the non-local model also has access to the basic properties of other mentions in the partial tree structure, such as head words. The

non-local features were selected with the same greedy forward strategy as the local features, starting from the optimized local feature sets.

7 Experimental Setup

We apply our model to the CoNLL 2012 Shared Task data, which includes a training, development, and test set split for three languages: Arabic, Chinese and English. We follow the *closed track* setting where systems may only be trained on the provided training data, with the exception of the English gender and number data compiled by Bergsma and Lin (2006). We use automatically extracted mentions using the same mention extraction procedure as Björkelund and Farkas (2012).

We evaluate our system using the CoNLL 2012 scorer, which computes several coreference metrics: MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), and $CEAF_e$ and $CEAF_m$ (Luo, 2005). We also report the CoNLL average (also known as MELA; Denis and Baldridge (2009)), i.e., the arithmetic mean of MUC, B^3, and $CEAF_e$. It should be noted that for B^3 and the CEAF metrics, multiple ways of handling *twinless mentions*[7] have been proposed (Rahman and Ng, 2009; Stoyanov et al., 2009). We use the most recent version of the CoNLL scorer (version 7), which implements the original definitions of these metrics.[8]

Our system is evaluated on the version of the data with automatic preprocessing information (e.g., predicted parse trees). Unless otherwise stated we use 25 iterations of perceptron training and a beam size of 20. We did not attempt to tune either of these parameters. We experiment with two feature sets for each language: the optimized local feature sets (denoted *local*), and the optimized local feature sets extended with non-local features (denoted *non-local*).

8 Results

Learning strategies. We begin by looking at the different learning strategies. Since early updates do not always make use of the complete documents during training, it can be expected that it will require either a very wide beam or more iterations to get up to par with the baseline learning algorithm. Figure 3 shows the CoNLL average on the English development set as a function of number of training iterations with two different beam sizes, 20 and 100, over the local and non-local feature sets. The figure shows that even after 50 iterations, early update falls short of the baseline, even when the early update system has access to more informative non-local features.[9]

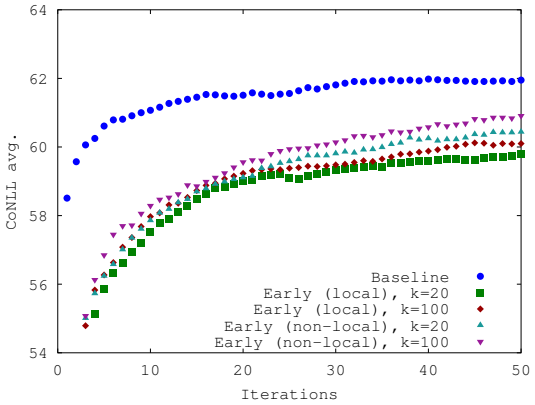


Figure 3: Comparing early update training with the baseline training algorithm.

In Figure 4 we compare early update with LaSO and delayed LaSO on the English development set. The left half uses the local feature set, and the right the extended non-local feature set. Recall that with only local features, delayed LaSO is equivalent to the baseline learning algorithm. As before, early update is considerably worse than other learning strategies. We also see that delayed LaSO outperforms LaSO, both with and without non-local features. Note that plain LaSO with non-local features only barely outperforms the delayed LaSO with only local features (i.e., the baseline), which indicates that only delayed LaSO is able to fully leverage non-local features. From these results we conclude that we are better off when the learning algorithm handles one document at a time, instead of getting feedback within documents.

Local vs. Non-local feature sets. Table 1 displays the differences in F-measures and CoNLL average between the local and non-local systems when applied to the development sets for each language. All metrics improve when more informative non-local features are added to the local feature set. Arabic and English show considerable improvements, and the CoNLL average increases

[7] i.e., mentions that appear in the prediction but not in gold, or the other way around

[8] Available at `http://conll.cemantix.org/2012/software.html`

[9] Although the Early systems still seem to show slight increases after 50 iterations, it needs a considerable number of iterations to catch up with the baseline – after 100 iterations the best early system is still more than half a point behind the baseline.

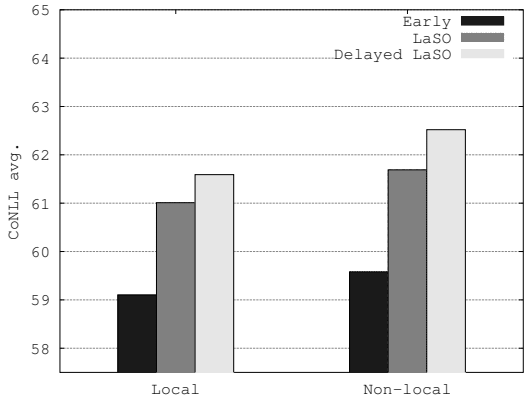


Figure 4: Comparison of learning algorithms evaluated on the English development set.

	MUC	B^3	$CEAF_m$	$CEAF_e$	CoNLL
Arabic					
local	47.33	42.51	49.71	46.49	45.44
non-local	49.31	43.52	50.96	47.18	46.67
Chinese					
local	65.84	57.94	62.23	57.05	60.27
non-local	66.4	57.99	62.37	57.12	60.5
English					
local	69.95	58.7	62.91	56.03	61.56
non-local	70.74	60.03	65.01	56.8	62.52

Table 1: Comparison of local and non-local feature sets on the development sets.

about one point. For Chinese the gains are generally not as pronounced, though the MUC metric goes up by more than half a point.

Final results. In Table 2 we compare the results of the non-local system (*This paper*) to the best results from the CoNLL 2012 Shared Task.[10] Specifically, this includes Fernandes et al.'s (2012) system for Arabic and English (denoted Fernandes), and Chen and Ng's (2012) system for Chinese (denoted C&N). For English we also compare it to the Berkeley system (Durrett and Klein, 2013), which, to our knowledge, is the best publicly available system for English coreference resolution (denoted D&K). As a general baseline, we also include Björkelund and Farkas' (2012) system (denoted B&F), which was the second best system in the shared task. For almost all metrics our system is significantly better than the best competitor. For a few metrics the best competitor outperforms our results for either precision or recall, but in terms of F-measures and the CoNLL average our system is the best for all languages.

[10]Thanks to Sameer Pradhan for providing us with the outputs of the other systems for significance testing.

9 Related Work

On the machine learning side Collins and Roark's (2004) work on the early update constitutes our starting point. The LaSO framework was introduced by Daumé III and Marcu (2005b), but has, to our knowledge, only been applied to the related task of entity detection and tracking (Daumé III and Marcu, 2005a). The theoretical motivation for early updates was only recently explained rigorously (Huang et al., 2012). The delayed LaSO update that we propose decomposes the prediction task of a complex structure into a number of subproblems, each of which guarantee *violation*, using Huang et al.'s (2012) terminology. We believe this is an interesting novelty, as it leverages the complete structures for every training instance during every iteration, and expect it to be applicable also to other structured prediction tasks.

Our approach also resembles imitation learning techniques such as SEARN (Daumé III et al., 2009) and DAGGER (Ross et al., 2011), where the search problem is reduced to a sequence of classification steps that guide the search algorithm through the search space. These frameworks, however, rely on the notion of an *expert policy* which provides an optimal decision at each point during search. In our context that would require antecedents for every mention to be given a priori, rather than using latent antecedents as we do.

Perceptrons for coreference. The perceptron has previously been used to train coreference resolvers either by casting the problem as a binary classification problem that considers pairs of mentions in isolation (Bengtson and Roth, 2008; Stoyanov et al., 2009; Chang et al., 2012, *inter alia*) or in the structured manner, where a clustering for an entire document is predicted in one go (Fernandes et al., 2012). However, none of these works use non-local features. Stoyanov and Eisner (2012) train an Easy-First coreference system with the perceptron to learn a sequence of join operations between arbitrary mentions in a document and accesses non-local features through previous merge operations in later stages. Culotta et al. (2007) also apply online learning in a first-order logic framework that enables non-local features, though using a greedy search algorithm.

Latent antecedents. The use of latent antecedents goes back to the work of Yu and Joachims (2009), although the idea of determining

	MUC			B^3			$CEAF_m$			$CEAF_e$			CoNLL
	Rec	Prec	F_1	Rec	Prec	F_1	Rec	Prec	F_1	Rec	Prec	F_1	avg.
						Arabic							
B&F	43.9	52.51	47.82	35.7	49.77	41.58	43.8	50.03	46.71	40.45	41.86	41.15	43.51
Fernandes	43.63	49.69	46.46	38.39	47.7	42.54	47.6	50.85	49.17	48.16	45.03	46.54	45.18
This paper	**47.53**	**53.3**	**50.25**	**44.14**	49.34	**46.6**	**50.94**	**55.19**	**52.98**	49.2	**49.45**	**49.33**	48.72
						Chinese							
B&F	58.72	58.49	58.61	49.17	53.2	51.11	56.68	51.86	54.14	55.36	41.8	47.63	52.45
C&N	59.92	64.69	62.21	51.76	60.26	55.69	59.58	60.45	60.02	**58.84**	51.61	54.99	57.63
This paper	**62.57**	**69.39**	**65.8**	**53.87**	**61.64**	**57.49**	58.75	**64.76**	**61.61**	54.65	**59.33**	**56.89**	60.06
						English							
B&F	65.23	70.1	67.58	49.51	60.69	54.47	56.93	59.51	58.19	51.34	49.14	59.21	57.42
Fernandes	65.83	75.91	70.51	51.55	65.19	57.58	57.48	65.93	61.42	50.82	57.28	53.86	60.65
D&K	66.58	74.94	70.51	53.2	**64.56**	58.33	59.19	66.23	62.51	52.9	58.06	55.36	61.4
This paper	**67.46**	74.3	**70.72**	**54.96**	62.71	**58.58**	**60.33**	**66.92**	**63.45**	52.27	**59.4**	55.61	61.63

Table 2: Comparison with other systems on the test sets. Bold numbers indicate significance at the $p < 0.05$ level between the best and the second best systems (according to the CoNLL average) using a Wilcoxon signed rank sum test. We refrain from significance tests on the CoNLL average, as it is an average over other F-measures.

meaningful antecedents for mentions can be traced back to Ng and Cardie (2002) who used a rule-based approach. Latent antecedents have recently gained popularity and were used by two systems in the CoNLL 2012 Shared Task, including the winning system (Fernandes et al., 2012; Chang et al., 2012). Durrett and Klein (2013) present a coreference resolver with latent antecedents that predicts clusterings over entire documents and fit a log-linear model with a custom task-specific loss function using AdaGrad (Duchi et al., 2011). Chang et al. (2013) use a max-margin approach to learn a pairwise model and rely on stochastic gradient descent to circumvent the costly operation of decoding the entire training set in order to compute the gradients and the latent antecedents. None of the aforementioned works use non-local features in their models, however.

Entity-mention models. Entity-mention models that compare a single mention to a (partial) cluster have been studied extensively and several works have evaluated non-local entity-level features (Luo et al., 2004; Yang et al., 2008; Rahman and Ng, 2009). Luo et al. (2004) also apply beam search at test time, but use a static assignment of antecedents and learns log-linear model using batch learning. Moreover, these works alter the basic feature definitions from their pairwise models when introducing entity-level features. This contrasts with our work, as our mention-pair model simply constitutes a special case of the non-local system.

10 Conclusion

We presented experiments with a coreference resolver that leverages non-local features to improve its performance. The application of non-local features requires the use of an approximate search algorithm to keep the problem tractable. We evaluated standard perceptron learning techniques for this setting both using early updates and LaSO. We found that the early update strategy is considerably worse than a local baseline, as it is unable to exploit all training data. LaSO resolves this issue by giving feedback within documents, but still underperforms compared to the baseline as it distorts the choice of latent antecedents.

We introduced a modification to LaSO, where updates are delayed until each document is processed. In the special case where only local features are used, this method coincides with standard structured perceptron learning that uses exact search. Moreover, it is also able to profit from non-local features resulting in improved performance. We evaluated our system on all three languages from the CoNLL 2012 Shared Task and present the best results to date on these data sets.

Acknowledgments

We are grateful to the anonymous reviewers as well as Christian Scheible and Wolfgang Seeker for comments on earlier versions of this paper. This research has been funded by the DFG via SFB 732, project D8.

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *In The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*, pages 563–566.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Eric Bengtson and Dan Roth. 2008. Understanding the value of features for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 294–303, Honolulu, Hawaii, October. Association for Computational Linguistics.

Shane Bergsma and Dekang Lin. 2006. Bootstrapping path-based pronoun resolution. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 33–40, Sydney, Australia, July. Association for Computational Linguistics.

Anders Björkelund and Richárd Farkas. 2012. Data-driven multilingual coreference resolution using resolver stacking. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 49–55, Jeju Island, Korea, July. Association for Computational Linguistics.

Bernd Bohnet, Simon Mille, Benoît Favre, and Leo Wanner. 2011. <stumaba >: From deep representation to surface. In *Proceedings of the Generation Challenges Session at the 13th European Workshop on Natural Language Generation*, pages 232–235, Nancy, France, September. Association for Computational Linguistics.

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 89–97, Beijing, China, August.

Kai-Wei Chang, Rajhans Samdani, Alla Rozovskaya, Mark Sammons, and Dan Roth. 2012. Illinois-coref: The ui system in the conll-2012 shared task. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 113–117, Jeju Island, Korea, July. Association for Computational Linguistics.

Kai-Wei Chang, Rajhans Samdani, and Dan Roth. 2013. A constrained latent variable model for coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 601–612, Seattle, Washington, USA, October. Association for Computational Linguistics.

Chen Chen and Vincent Ng. 2012. Combining the best of two worlds: A hybrid approach to multilingual coreference resolution. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 56–63, Jeju Island, Korea, July. Association for Computational Linguistics.

Yoeng-jin Chu and Tseng-hong Liu. 1965. On the shortest aborescence of a directed graph. *Science Sinica*, 14:1396–1400.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 111–118, Barcelona, Spain, July.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8. Association for Computational Linguistics, July.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive–aggressive algorithms. *Journal of Machine Learning Reseach*, 7:551–585, March.

Aron Culotta, Michael Wick, and Andrew McCallum. 2007. First-order probabilistic models for coreference resolution. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 81–88, Rochester, New York, April. Association for Computational Linguistics.

Hal Daumé III and Daniel Marcu. 2005a. A large-scale exploration of effective global features for a joint entity detection and tracking model. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 97–104, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Hal Daumé III and Daniel Marcu. 2005b. Learning as search optimization: approximate large margin methods for structured prediction. In *ICML*, pages 169–176.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning*, 75(3):297–325.

Pascal Denis and Jason Baldridge. 2009. Global Joint Models for Coreference Resolution and Named Entity Classification. In *Procesamiento del Lenguaje Natural 42*, pages 87–96, Barcelona: SEPLN.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1971–1982,

Seattle, Washington, USA, October. Association for Computational Linguistics.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71(B):233–240.

Eraldo Fernandes, Cícero dos Santos, and Ruy Milidiú. 2012. Latent structure perceptron with feature induction for unrestricted coreference resolution. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 41–48, Jeju Island, Korea, July. Association for Computational Linguistics.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–151, Montréal, Canada, June. Association for Computational Linguistics.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*, pages 586–594, Columbus, Ohio, June. Association for Computational Linguistics.

Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the bell tree. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, pages 135–142, Barcelona, Spain, July.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 25–32, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 104–111, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Vincent Ng. 2010. Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1396–1411, Uppsala, Sweden, July. Association for Computational Linguistics.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40, Jeju Island, Korea, July. Association for Computational Linguistics.

Altaf Rahman and Vincent Ng. 2009. Supervised models for coreference resolution. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 968–977, Singapore, August. Association for Computational Linguistics.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado, June. Association for Computational Linguistics.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, pages 627–635.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

Veselin Stoyanov and Jason Eisner. 2012. Easy-first coreference resolution. In *Proceedings of COLING 2012*, pages 2519–2534, Mumbai, India, December.

Veselin Stoyanov, Nathan Gilbert, Claire Cardie, and Ellen Riloff. 2009. Conundrums in noun phrase coreference resolution: Making sense of the state-of-the-art. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 656–664, Suntec, Singapore, August. Association for Computational Linguistics.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model theoretic coreference scoring scheme. In *Proceedings MUC-6*, pages 45–52, Columbia, Maryland.

Xiaofeng Yang, Jian Su, Jun Lang, Chew Lim Tan, Ting Liu, and Sheng Li. 2008. An entity-mention model for coreference resolution with inductive logic programming. In *Proceedings of ACL-08: HLT*, pages 843–851, Columbus, Ohio, June. Association for Computational Linguistics.

Chun-Nam Yu and T. Joachims. 2009. Learning structural svms with latent variables. In *International Conference on Machine Learning (ICML)*.

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

Multilingual Models for Compositional Distributed Semantics

Karl Moritz Hermann and **Phil Blunsom**
Department of Computer Science
University of Oxford
Oxford, OX1 3QD, UK
{karl.moritz.hermann,phil.blunsom}@cs.ox.ac.uk

Abstract

We present a novel technique for learning semantic representations, which extends the distributional hypothesis to multilingual data and joint-space embeddings. Our models leverage parallel data and learn to strongly align the embeddings of semantically equivalent sentences, while maintaining sufficient distance between those of dissimilar sentences. The models do not rely on word alignments or any syntactic information and are successfully applied to a number of diverse languages. We extend our approach to learn semantic representations at the document level, too. We evaluate these models on two cross-lingual document classification tasks, outperforming the prior state of the art. Through qualitative analysis and the study of pivoting effects we demonstrate that our representations are semantically plausible and can capture semantic relationships across languages without parallel data.

1 Introduction

Distributed representations of words provide the basis for many state-of-the-art approaches to various problems in natural language processing today. Such word embeddings are naturally richer representations than those of symbolic or discrete models, and have been shown to be able to capture both syntactic and semantic information. Successful applications of such models include language modelling (Bengio et al., 2003), paraphrase detection (Erk and Padó, 2008), and dialogue analysis (Kalchbrenner and Blunsom, 2013).

Within a monolingual context, the distributional hypothesis (Firth, 1957) forms the basis of most approaches for learning word representations. In this work, we extend this hypothesis to multilingual data and joint-space embeddings. We present a novel unsupervised technique for learning semantic representations that leverages parallel corpora and employs semantic transfer through compositional representations. Unlike most methods for learning word representations, which are restricted to a single language, our approach learns to represent meaning across languages in a shared multilingual semantic space.

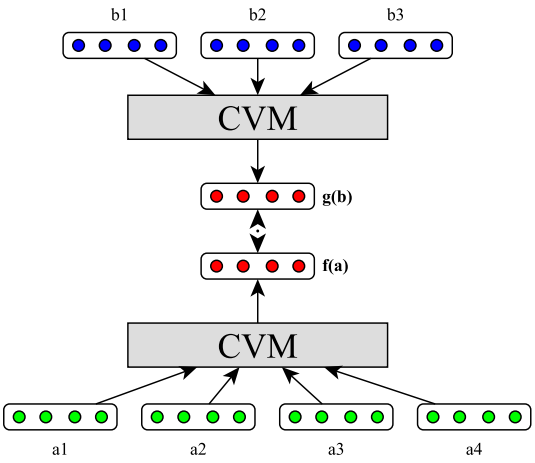


Figure 1: Model with parallel input sentences a and b. The model minimises the distance between the sentence level encoding of the bitext. Any composition functions (CVM) can be used to generate the compositional sentence level representations.

We present experiments on two corpora. First, we show that for cross-lingual document classification on the Reuters RCV1/RCV2 corpora (Lewis et al., 2004), we outperform the prior state of the art (Klementiev et al., 2012). Second, we also present classification results on a massively multilingual corpus which we derive from the TED corpus (Cettolo et al., 2012). The results on this task, in comparison with a number of strong baselines, further demonstrate the relevance of our approach and the success of our method in learning multilingual semantic representations over a wide range of languages.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 58–68,
Baltimore, Maryland, USA, June 23-25 2014.

2 Overview

Distributed representation learning describes the task of learning continuous representations for discrete objects. Here, we focus on learning semantic representations and investigate how the use of multilingual data can improve learning such representations at the word and higher level. We present a model that learns to represent each word in a lexicon by a continuous vector in $\mathbb{R}^d$. Such distributed representations allow a model to share meaning between similar words, and have been used to capture semantic, syntactic and morphological content (Collobert and Weston, 2008; Turian et al., 2010, *inter alia*).

We describe a multilingual objective function that uses a noise-contrastive update between semantic representations of different languages to learn these word embeddings. As part of this, we use a compositional vector model (CVM, henceforth) to compute semantic representations of sentences and documents. A CVM learns semantic representations of larger syntactic units given the semantic representations of their constituents (Clark and Pulman, 2007; Mitchell and Lapata, 2008; Baroni and Zamparelli, 2010; Grefenstette and Sadrzadeh, 2011; Socher et al., 2012; Hermann and Blunsom, 2013, *inter alia*).

A key difference between our approach and those listed above is that we only require sentence-aligned parallel data in our otherwise unsupervised learning function. This removes a number of constraints that normally come with CVM models, such as the need for syntactic parse trees, word alignment or annotated data as a training signal. At the same time, by using multiple CVMs to transfer information between languages, we enable our models to capture a broader semantic context than would otherwise be possible.

The idea of extracting semantics from multilingual data stems from prior work in the field of semantic grounding. Language acquisition in humans is widely seen as grounded in sensory-motor experience (Bloom, 2001; Roy, 2003). Based on this idea, there have been some attempts at using multi-modal data for learning better vector representations of words (e.g. Srivastava and Salakhutdinov (2012)). Such methods, however, are not easily scalable across languages or to large amounts of data for which no secondary or tertiary representation might exist.

Parallel data in multiple languages provides an alternative to such secondary representations, as parallel texts share their semantics, and thus one language can be used to ground the other. Some work has exploited this idea for transferring linguistic knowledge into low-resource languages or to learn distributed representations at the word level (Klementiev et al., 2012; Zou et al., 2013; Lauly et al., 2013, *inter alia*). So far almost all of this work has been focused on learning multilingual representations at the word level. As distributed representations of larger expressions have been shown to be highly useful for a number of tasks, it seems to be a natural next step to attempt to induce these, too, cross-lingually.

3 Approach

Most prior work on learning compositional semantic representations employs parse trees on their training data to structure their composition functions (Socher et al., 2012; Hermann and Blunsom, 2013, *inter alia*). Further, these approaches typically depend on specific *semantic* signals such as sentiment- or topic-labels for their objective functions. While these methods have been shown to work in some cases, the need for parse trees and annotated data limits such approaches to resource-fortunate languages. Our novel method for learning compositional vectors removes these requirements, and as such can more easily be applied to low-resource languages.

Specifically, we attempt to learn semantics from multilingual data. The idea is that, given enough parallel data, a shared representation of two parallel sentences would be forced to capture the common elements between these two sentences. What parallel sentences share, of course, are their semantics. Naturally, different languages express meaning in different ways. We utilise this diversity to abstract further from mono-lingual surface realisations to deeper semantic representations. We exploit this semantic similarity across languages by defining a bilingual (and trivially multilingual) energy as follows.

Assume two functions $f : X \to \mathbb{R}^d$ and $g : Y \to \mathbb{R}^d$, which map sentences from languages x and y onto distributed semantic representations in $\mathbb{R}^d$. Given a parallel corpus C, we then define the energy of the model given two sentences $(a, b) \in C$ as:

$$E_{bi}(a, b) = \|f(a) - g(b)\|^2 \qquad (1)$$

We want to minimize E_{bi} for all semantically equivalent sentences in the corpus. In order to prevent the model from degenerating, we further introduce a noise-constrastive large-margin update which ensures that the representations of non-aligned sentences observe a certain margin from each other. For every pair of parallel sentences (a, b) we sample a number of additional sentence pairs $(\cdot, n) \in C$, where n—with high probability—is not semantically equivalent to a. We use these noise samples as follows:

$$E_{hl}(a, b, n) = [m + E_{bi}(a, b) - E_{bi}(a, n)]_+$$

where $[x]_+ = max(x, 0)$ denotes the standard hinge loss and m is the margin. This results in the following objective function:

$$J(\theta) = \sum_{(a,b)\in\mathcal{C}} \left(\sum_{i=1}^{k} E_{hl}(a, b, n_i) + \frac{\lambda}{2}\|\theta\|^2 \right) \quad (2)$$

where θ is the set of all model variables.

3.1 Two Composition Models

The objective function in Equation 2 could be coupled with any two given vector composition functions f, g from the literature. As we aim to apply our approach to a wide range of languages, we focus on composition functions that do not require any syntactic information. We evaluate the following two composition functions.

The first model, ADD, represents a sentence by the sum of its word vectors. This is a distributed bag-of-words approach as sentence ordering is not taken into account by the model.

Second, the BI model is designed to capture bigram information, using a non-linearity over bigram pairs in its composition function:

$$f(x) = \sum_{i=1}^{n} \tanh\left(x_{i-1} + x_i\right) \quad (3)$$

The use of a non-linearity enables the model to learn interesting interactions between words in a document, which the bag-of-words approach of ADD is not capable of learning. We use the hyperbolic tangent as activation function.

3.2 Document-level Semantics

For a number of tasks, such as topic modelling, representations of objects beyond the sentence level are required. While most approaches to compositional distributed semantics end at the word level, our model extends to document-level learning quite naturally, by recursively applying the composition and objective function (Equation 2) to compose sentences into documents. This is achieved by first computing semantic representations for each sentence in a document. Next, these representations are used as inputs in a higher-level CVM, computing a semantic representation of a document (Figure 2).

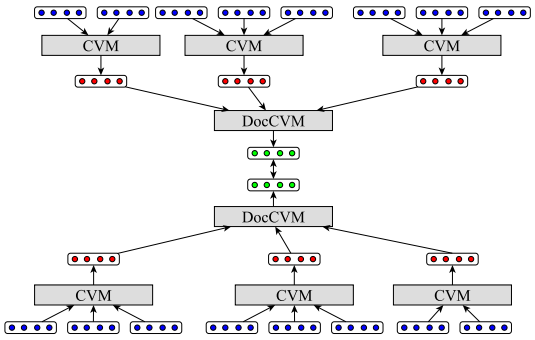


Figure 2: Description of a parallel document-level compositional vector model (DOC). The model recursively computes semantic representations for each sentence of a document and then for the document itself, treating the sentence vectors as inputs for a second CVM.

This recursive approach integrates document-level representations into the learning process. We can thus use corpora of parallel documents—regardless of whether they are sentence aligned or not—to propagate a semantic signal back to the individual words. If sentence alignment is available, of course, the document-signal can simply be combined with the sentence-signal, as we did with the experiments described in §5.3.

This concept of learning compositional representations for documents contrasts with prior work (Socher et al., 2011; Klementiev et al., 2012, *inter alia*) who rely on summing or averaging sentence-vectors if representations beyond the sentence-level are required for a particular task.

We evaluate the models presented in this paper both with and without the document-level signal. We refer to the individual models used as ADD and BI if used without, and as DOC/ADD and DOC/BI is used with the additional document composition function and error signal.

4 Corpora

We use two corpora for learning semantic representations and performing the experiments described in this paper.

The Europarl corpus v7[1] (Koehn, 2005) was used during initial development and testing of our approach, as well as to learn the representations used for the Cross-Lingual Document Classification task described in §5.2. We considered the English-German and English-French language pairs from this corpus. From each pair the final 100,000 sentences were reserved for development.

Second, we developed a massively multilingual corpus based on the TED corpus[2] for IWSLT 2013 (Cettolo et al., 2012). This corpus contains English transcriptions and multilingual, sentence-aligned translations of talks from the TED conference. While the corpus is aimed at machine translation tasks, we use the keywords associated with each talk to build a subsidiary corpus for multilingual document classification as follows.[3]

The development sections provided with the IWSLT 2013 corpus were again reserved for development. We removed approximately 10 percent of the training data in each language to create a test corpus (all talks with $id \geq 1{,}400$). The new training corpus consists of a total of 12,078 parallel documents distributed across 12 language pairs[4]. In total, this amounts to 1,678,219 non-English sentences (the number of unique English sentences is smaller as many documents are translated into multiple languages and thus appear repeatedly in the corpus). Each document (talk) contains one or several keywords. We used the 15 most frequent keywords for the topic classification experiments described in section §5.3.

Both corpora were pre-processed using the set of tools provided by cdec[5] for tokenizing and lowercasing the data. Further, all empty sentences and their translations were removed from the corpus.

5 Experiments

We report results on two experiments. First, we replicate the cross-lingual document classification task of Klementiev et al. (2012), learning distributed representations on the Europarl corpus and evaluating on documents from the Reuters RCV1/RCV2 corpora. Subsequently, we design a multi-label classification task using the TED corpus, both for training and evaluating. The use of a wider range of languages in the second experiments allows us to better evaluate our models' capabilities in learning a shared multilingual semantic representation. We also investigate the learned embeddings from a qualitative perspective in §5.4.

5.1 Learning

All model weights were randomly initialised using a Gaussian distribution (μ=0, σ^2=0.1). We used the available development data to set our model parameters. For each positive sample we used a number of noise samples ($k \in \{1, 10, 50\}$), randomly drawn from the corpus at each training epoch. All our embeddings have dimensionality d=128, with the margin set to m=d.[6] Further, we use L2 regularization with λ=1 and step-size in $\{0.01, 0.05\}$. We use 100 iterations for the RCV task, 500 for the TED single and 5 for the joint corpora. We use the adaptive gradient method, AdaGrad (Duchi et al., 2011), for updating the weights of our models, in a mini-batch setting ($b \in \{10, 50\}$). All settings, our model implementation and scripts to replicate our experiments are available at `http://www.karlmoritz.com/`.

5.2 RCV1/RCV2 Document Classification

We evaluate our models on the cross-lingual document classification (CLDC, henceforth) task first described in Klementiev et al. (2012). This task involves learning language independent embeddings which are then used for document classification across the English-German language pair. For this, CLDC employs a particular kind of supervision, namely using supervised training data in one language and evaluating without further supervision in another. Thus, CLDC can be used to establish whether our learned representations are semantically useful across multiple languages.

We follow the experimental setup described in Klementiev et al. (2012), with the exception that we learn our embeddings using solely the Europarl data and use the Reuters corpora only during for classifier training and testing. Each document in the classification task is represented by the average of the d-dimensional representations of all its sentences. We train the multiclass classifier using an averaged perceptron (Collins, 2002) with the same settings as in Klementiev et al. (2012).

[1] `http://www.statmt.org/europarl/`

[2] `https://wit3.fbk.eu/`

[3] `http://www.clg.ox.ac.uk/tedcldc/`

[4] English to Arabic, German, French, Spanish, Italian, Dutch, Polish, Brazilian Portuguese, Romanian, Russian and Turkish. Chinese, Farsi and Slowenian were removed due to the small size of those datasets.

[5] `http://cdec-decoder.org/`

[6] On the RCV task we also report results for d=40 which matches the dimensionality of Klementiev et al. (2012).

Model	en → de	de → en
Majority Class	46.8	46.8
Glossed	65.1	68.6
MT	68.1	67.4
I-Matrix	77.6	71.1
$dim = 40$		
ADD	83.7	71.4
ADD+	86.2	76.9
BI	83.4	69.2
BI+	86.9	74.3
$dim = 128$		
ADD	86.4	74.7
ADD+	87.7	77.5
BI	86.1	79.0
BI+	**88.1**	**79.2**

Table 1: Classification accuracy for training on English and German with 1000 labeled examples on the RCV corpus. Cross-lingual compositional representations (ADD, BI and their multilingual extensions), I-Matrix (Klementiev et al., 2012) translated (MT) and glossed (Glossed) word baselines, and the majority class baseline. The baseline results are from Klementiev et al. (2012).

We present results from four models. The ADD model is trained on 500k sentence pairs of the English-German parallel section of the Europarl corpus. The ADD+ model uses an additional 500k parallel sentences from the English-French corpus, resulting in one million English sentences, each paired up with either a German or a French sentence, with BI and BI+ trained accordingly. The motivation behind ADD+ and BI+ is to investigate whether we can learn better embeddings by introducing additional data from other languages. A similar idea exists in machine translation where English is frequently used to pivot between other languages (Cohn and Lapata, 2007).

The actual CLDC experiments are performed by training on English and testing on German documents and vice versa. Following prior work, we use varying sizes between 100 and 10,000 documents when training the multiclass classifier. The results of this task across training sizes are in Figure 3. Table 1 shows the results for training on 1,000 documents compared with the results published in Klementiev et al. (2012). Our models outperform the prior state of the art, with the BI models performing slightly better than the ADD models. As the relative results indicate, the addition of a second language improves model performance. It it interesting to note that results improve in both directions of the task, even though no additional German data was used for the '+' models.

5.3 TED Corpus Experiments

Here we describe our experiments on the TED corpus, which enables us to scale up to multilingual learning. Consisting of a large number of relatively short and parallel documents, this corpus allows us to evaluate the performance of the DOC model described in §3.2.

We use the training data of the corpus to learn distributed representations across 12 languages. Training is performed in two settings. In the *single* mode, vectors are learnt from a single language pair (en-X), while in the *joint* mode vector-learning is performed on all parallel sub-corpora simultaneously. This setting causes words from all languages to be embedded in a single semantic space.

First, we evaluate the effect of the document-level error signal (DOC, described in §3.2), as well as whether our multilingual learning method can extend to a larger variety of languages. We train DOC models, using both ADD and BI as CVM (DOC/ADD, DOC/BI), both in the *single* and *joint* mode. For comparison, we also train ADD and DOC models without the document-level error signal. The resulting document-level representations are used to train classifiers (system and settings as in §5.2) for each language, which are then evaluated in the paired language. In the English case we train twelve individual classifiers, each using the training data of a single language pair only. As described in §4, we use 15 keywords for the classification task. Due to space limitations, we report cumulative results in the form of F1-scores throughout this paper.

MT System We develop a machine translation baseline as follows. We train a machine translation tool on the parallel training data, using the development data of each language pair to optimize the translation system. We use the cdec decoder (Dyer et al., 2010) with default settings for this purpose. With this system we translate the test data, and then use a Naïve Bayes classifier[7] for the actual experiments. To exemplify, this means the $de{\rightarrow}ar$ result is produced by training a translation system from Arabic to German. The Arabic test set is translated into German. A classifier is then trained

[7]We use the implementation in Mallet (McCallum, 2002)

Setting	Languages										
	Arabic	German	Spanish	French	Italian	Dutch	Polish	Pt-Br	Roman.	Russian	Turkish
$en \rightarrow$ L2											
MT System	**0.429**	**0.465**	**0.518**	**0.526**	**0.514**	**0.505**	**0.445**	**0.470**	**0.493**	0.432	0.409
ADD *single*	0.328	0.343	0.401	0.275	0.282	0.317	0.141	0.227	0.282	0.338	0.241
BI *single*	0.375	0.360	0.379	0.431	0.465	0.421	0.435	0.329	0.426	0.423	**0.481**
DOC/ADD *single*	0.410	0.424	0.383	0.476	0.485	0.264	0.402	0.354	0.418	0.448	0.452
DOC/BI *single*	0.389	0.428	0.416	0.445	0.473	0.219	0.403	0.400	0.467	0.421	0.457
DOC/ADD *joint*	0.392	0.405	0.443	0.447	0.475	0.453	0.394	0.409	0.446	**0.476**	0.417
DOC/BI *joint*	0.372	0.369	0.451	0.429	0.404	0.433	0.417	0.399	0.453	0.439	0.418
L2 $\rightarrow$ en											
MT System	0.448	0.469	**0.486**	0.358	**0.481**	0.463	**0.460**	0.374	**0.486**	0.404	0.441
ADD *single*	0.380	0.337	0.446	0.293	0.357	0.295	0.327	0.235	0.293	0.355	0.375
BI *single*	0.354	0.411	0.344	0.426	0.439	0.428	0.443	0.357	0.426	0.442	0.403
DOC/ADD *single*	**0.452**	**0.476**	0.422	0.464	0.461	0.251	0.400	0.338	0.407	**0.471**	0.435
DOC/BI *single*	0.406	0.442	0.365	**0.479**	0.460	0.235	0.393	0.380	0.426	0.467	**0.477**
DOC/ADD *joint*	0.396	0.388	0.399	0.415	0.461	**0.478**	0.352	**0.399**	0.412	0.343	0.343
DOC/BI *joint*	0.343	0.375	0.369	0.419	0.398	0.438	0.353	0.391	0.430	0.375	0.388

Table 2: F1-scores for the TED document classification task for individual languages. Results are reported for both directions (training on English, evaluating on L2 and vice versa). Bold indicates best result, underline best result amongst the vector-based systems.

Training Language	Test Language										
	Arabic	German	Spanish	French	Italian	Dutch	Polish	Pt-Br	Rom'n	Russian	Turkish
Arabic		0.378	0.436	0.432	0.444	0.438	0.389	0.425	0.420	0.446	0.397
German	0.368		0.474	0.460	0.464	0.440	0.375	0.417	0.447	0.458	0.443
Spanish	0.353	0.355		0.420	0.439	0.435	0.415	0.390	0.424	0.427	0.382
French	0.383	0.366	0.487		0.474	0.429	0.403	0.418	0.458	0.415	0.398
Italian	0.398	0.405	0.461	0.466		0.393	0.339	0.347	0.376	0.382	0.352
Dutch	0.377	0.354	0.463	0.464	0.460		0.405	0.386	0.415	0.407	0.395
Polish	0.359	0.386	0.449	0.444	0.430	0.441		0.401	0.434	0.398	0.408
Portuguese	0.391	0.392	0.476	0.447	0.486	0.458	0.403		0.457	0.431	0.431
Romanian	0.416	0.320	0.473	0.476	0.460	0.434	0.416	0.433		0.444	0.402
Russian	0.372	0.352	0.492	0.427	0.438	0.452	0.430	0.419	0.441		0.447
Turkish	0.376	0.352	0.479	0.433	0.427	0.423	0.439	0.367	0.434	0.411	

Table 3: F1-scores for TED corpus document classification results when training and testing on two languages that do not share any parallel data. We train a DOC/ADD model on all en-L2 language pairs together, and then use the resulting embeddings to train document classifiers in each language. These classifiers are subsequently used to classify data from all other languages.

Setting	Languages											
	English	Arabic	German	Spanish	French	Italian	Dutch	Polish	Pt-Br	Roman.	Russian	Turkish
Raw Data NB	0.481	0.469	0.471	0.526	0.532	0.524	0.522	0.415	0.465	0.509	0.465	0.513
Senna	0.400											
Polyglot	0.382	0.416	0.270	0.418	0.361	0.332	0.228	0.323	0.194	0.300	0.402	0.295
single Setting												
DOC/ADD	0.462	0.422	0.429	0.394	0.481	0.458	0.252	0.385	0.363	0.431	0.471	0.435
DOC/BI	0.474	0.432	0.362	0.336	0.444	0.469	0.197	0.414	0.395	0.445	0.436	0.428
joint Setting												
DOC/ADD	0.475	0.371	0.386	0.472	0.451	0.398	0.439	0.304	0.394	0.453	0.402	0.441
DOC/BI	0.378	0.329	0.358	0.472	0.454	0.399	0.409	0.340	0.431	0.379	0.395	0.435

Table 4: F1-scores on the TED corpus document classification task when training and evaluating on the same language. Baseline embeddings are Senna (Collobert et al., 2011) and Polyglot (Al-Rfou' et al., 2013).

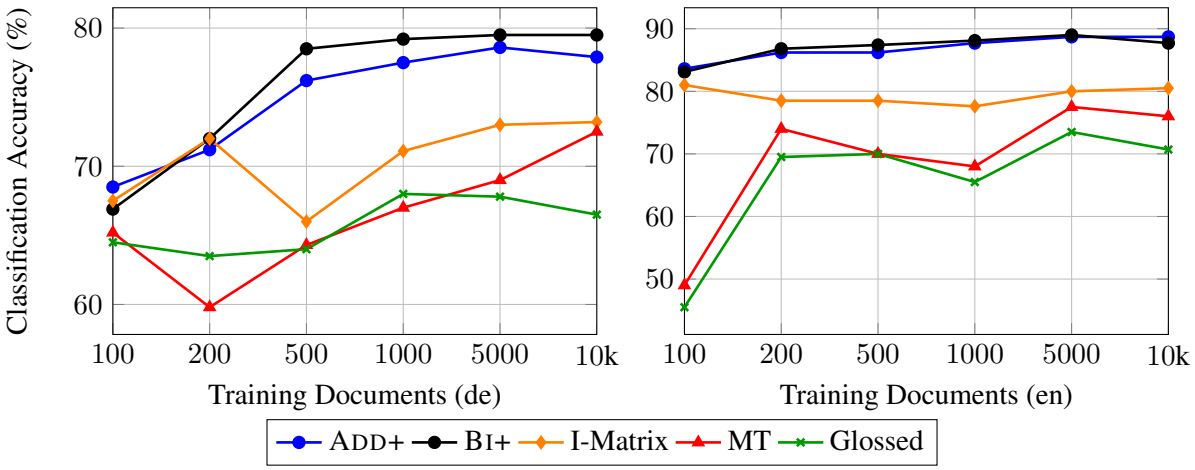


Figure 3: Classification accuracy for a number of models (see Table 1 for model descriptions). The left chart shows results for these models when trained on German data and evaluated on English data, the right chart vice versa.

on the German training data and evaluated on the translated Arabic. While we developed this system as a baseline, it must be noted that the classifier of this system has access to significantly more information (all words in the document) as opposed to our models (one embedding per document), and we do not expect to necessarily beat this system.

The results of this experiment are in Table 2. When comparing the results between the ADD model and the models trained using the document-level error signal, the benefit of this additional signal becomes clear. The *joint* training mode leads to a relative improvement when training on English data and evaluating in a second language. This suggests that the *joint* mode improves the quality of the English embeddings more than it affects the L2-embeddings. More surprising, perhaps, is the relative performance between the ADD and BI composition functions, especially when compared to the results in §5.2, where the BI models relatively consistently performed better. We suspect that the better performance of the additive composition function on this task is related to the smaller amount of training data available which could cause sparsity issues for the bigram model.

As expected, the MT system slightly outperforms our models on most language pairs. However, the overall performance of the models is comparable to that of the MT system. Considering the relative amount of information available during the classifier training phase, this indicates that our learned representations are semantically useful, capturing almost the same amount of information as available to the Naïve Bayes classifier.

We next investigate linguistic transfer across languages. We re-use the embeddings learned with the DOC/ADD *joint* model from the previous experiment for this purpose, and train classifiers on all non-English languages using those embeddings. Subsequently, we evaluate their performance in classifying documents in the remaining languages. Results for this task are in Table 3. While the results across language-pairs might not be very insightful, the overall good performance compared with the results in Table 2 implies that we learnt semantically meaningful vectors and in fact a joint embedding space across thirteen languages.

In a third evaluation (Table 4), we apply the embeddings learnt with out models to a monolingual classification task, enabling us to compare with prior work on distributed representation learning. In this experiment a classifier is trained in one language and then evaluated in the same. We again use a Naïve Bayes classifier on the raw data to establish a reasonable upper bound.

We compare our embeddings with the SENNA embeddings, which achieve state of the art performance on a number of tasks (Collobert et al., 2011). Additionally, we use the Polyglot embeddings of Al-Rfou' et al. (2013), who published word embeddings across 100 languages, including all languages considered in this paper. We represent each document by the mean of its word vectors and then apply the same classifier training and testing regime as with our models. Even though both of these sets of embeddings were trained on much larger datasets than ours, our models outperform these baselines on all languages—even outperforming the Naïve Bayes system on on several

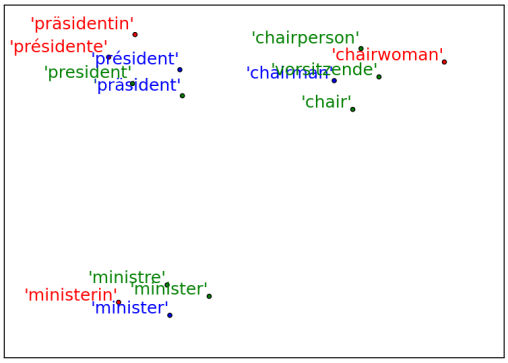


Figure 4: t-SNE projections for a number of English, French and German words as represented by the BI+ model. Even though the model did not use any parallel French-German data during training, it learns semantic similarity between these two languages using English as a pivot, and semantically clusters words across all languages.

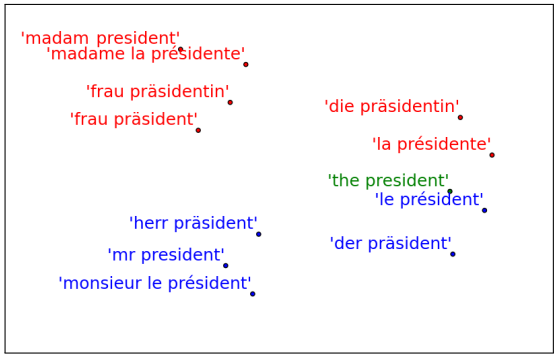


Figure 5: t-SNE projections for a number of short phrases in three languages as represented by the BI+ model. The projection demonstrates linguistic transfer through a pivot by. It separates phrases by gender (red for female, blue for male, and green for neutral) and aligns matching phrases across languages.

languages. While this may partly be attributed to the fact that our vectors were learned on in-domain data, this is still a very positive outcome.

5.4 Linguistic Analysis

While the classification experiments focused on establishing the semantic content of the sentence level representations, we also want to briefly investigate the induced word embeddings. We use the BI+ model trained on the Europarl corpus for this purpose. Figure 4 shows the t-SNE projections for a number of English, French and German words. Even though the model did not use any parallel French-German data during training, it still managed to learn semantic word-word similarity across these two languages.

Going one step further, Figure 5 shows t-SNE projections for a number of short phrases in these three languages. We use the English *the president* and gender-specific expressions *Mr President* and *Madam President* as well as gender-specific equivalents in French and German. The projection demonstrates a number of interesting results: First, the model correctly clusters the words into three groups, corresponding to the three English forms and their associated translations. Second, a separation between genders can be observed, with male forms on the bottom half of the chart and female forms on the top, with the neutral *the president* in the vertical middle. Finally, if we assume a horizontal line going through *the president*, this line could be interpreted as a "gender divide", with male and female versions of one expression mirroring each other on that line. In the case of *the president* and its translations, this effect becomes even clearer, with the neutral English expression being projected close to the mid-point between each other language's gender-specific versions.

These results further support our hypothesis that the bilingual contrastive error function can learn semantically plausible embeddings and furthermore, that it can abstract away from mono-lingual surface realisations into a shared semantic space across languages.

6 Related Work

Distributed Representations Distributed representations can be learned through a number of approaches. In their simplest form, distributional information from large corpora can be used to learn embeddings, where the words appearing within a certain window of the target word are used to compute that word's embedding. This is related to topic-modelling techniques such as LSA (Dumais et al., 1988), LSI, and LDA (Blei et al., 2003), but these methods use a document-level context, and tend to capture the topics a word is used in rather than its more immediate syntactic context.

Neural language models are another popular approach for inducing distributed word representations (Bengio et al., 2003). They have received a lot of attention in recent years (Collobert and Weston, 2008; Mnih and Hinton, 2009; Mikolov et al., 2010, *inter alia*) and have achieved state of the art performance in language modelling. Collobert et al. (2011) further popularised using neural network architectures for learning word embeddings from large amounts of largely unlabelled data by showing the embeddings can then be used to improve standard supervised tasks.

Unsupervised word representations can easily be plugged into a variety of NLP related tasks. Tasks, where the use of distributed representations has resulted in improvements include topic modelling (Blei et al., 2003) or named entity recognition (Turian et al., 2010; Collobert et al., 2011).

Compositional Vector Models For a number of important problems, semantic representations of individual words do not suffice, but instead a semantic representation of a larger structure—e.g. a phrase or a sentence—is required. Self-evidently, sparsity prevents the learning of such representations using the same collocational methods as applied to the word level. Most literature instead focuses on learning composition functions that represent the semantics of a larger structure as a function of the representations of its parts.

Very simple composition functions have been shown to suffice for tasks such as judging bigram semantic similarity (Mitchell and Lapata, 2008). More complex composition functions using matrix-vector composition, convolutional neural networks or tensor composition have proved useful in tasks such as sentiment analysis (Socher et al., 2011; Hermann and Blunsom, 2013), relational similarity (Turney, 2012) or dialogue analysis (Kalchbrenner and Blunsom, 2013).

Multilingual Representation Learning Most research on distributed representation induction has focused on single languages. English, with its large number of annotated resources, has enjoyed most attention. However, there exists a corpus of prior work on learning multilingual embeddings or on using parallel data to transfer linguistic information across languages. One has to differentiate between approaches such as Al-Rfou' et al. (2013), that learn embeddings across a large variety of languages and models such as ours, that learn joint embeddings, that is a projection into a shared semantic space across multiple languages.

Related to our work, Yih et al. (2011) proposed S2Nets to learn joint embeddings of tf-idf vectors for comparable documents. Their architecture optimises the cosine similarity of documents, using relative semantic similarity scores during learning. More recently, Lauly et al. (2013) proposed a bag-of-words autoencoder model, where the bag-of-words representation in one language is used to train the embeddings in another. By placing their vocabulary in a binary branching tree, the probabilistic setup of this model is similar to that of Mnih and Hinton (2009). Similarly, Sarath Chandar et al. (2013) train a cross-lingual encoder, where an autoencoder is used to recreate words in two languages in parallel. This is effectively the linguistic extension of Ngiam et al. (2011), who used a similar method for audio and video data. Hermann and Blunsom (2014) propose a large-margin learner for multilingual word representations, similar to the basic additive model proposed here, which, like the approaches above, relies on a bag-of-words model for sentence representations.

Klementiev et al. (2012), our baseline in §5.2, use a form of multi-agent learning on word-aligned parallel data to transfer embeddings from one language to another. Earlier work, Haghighi et al. (2008), proposed a method for inducing bilingual lexica using monolingual feature representations and a small initial lexicon to bootstrap with. This approach has recently been extended by Mikolov et al. (2013a), Mikolov et al. (2013b), who developed a method for learning transformation matrices to convert semantic vectors of one language into those of another. Is was demonstrated that this approach can be applied to improve tasks related to machine translation. Their CBOW model is also worth noting for its similarities to the ADD composition function used here. Using a slightly different approach, Zou et al. (2013), also learned bilingual embeddings for machine translation.

7 Conclusion

To summarize, we have presented a novel method for learning multilingual word embeddings using parallel data in conjunction with a multilingual objective function for compositional vector models. This approach extends the distributional hypothesis to multilingual joint-space representations. Coupled with very simple composition functions, vectors learned with this method outperform the state of the art on the task of cross-lingual document classification. Further experiments and analysis support our hypothesis that bilingual signals are a useful tool for learning distributed representations by enabling models to abstract away from mono-lingual surface realisations into a deeper semantic space.

Acknowledgements

This work was supported by a Xerox Foundation Award and EPSRC grant number EP/K036580/1.

References

R. Al-Rfou', B. Perozzi, and S. Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of CoNLL*.

M. Baroni and R. Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of EMNLP*.

Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, March.

D. M. Blei, A. Y. Ng, and M. I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

P. Bloom. 2001. Precis of how children learn the meanings of words. *Behavioral and Brain Sciences*, 24:1095–1103.

M. Cettolo, C. Girardi, and M. Federico. 2012. Wit^3: Web inventory of transcribed and translated talks. In *Proceedings of EAMT*.

S. Clark and S. Pulman. 2007. Combining symbolic and distributional models of meaning. In *Proceedings of AAAI Spring Symposium on Quantum Interaction*. AAAI Press.

T. Cohn and M. Lapata. 2007. Machine translation by triangulation: Making effective use of multi-parallel corpora. In *Proceedings of ACL*.

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of ACL-EMNLP*.

R. Collobert and J. Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

J. Duchi, E. Hazan, and Y. Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, July.

S. T. Dumais, G. W. Furnas, T. K. Landauer, S. Deerwester, and R. Harshman. 1988. Using latent semantic analysis to improve access to textual information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*.

C. Dyer, A. Lopez, J. Ganitkevitch, J. Weese, F. Ture, P. Blunsom, H. Setiawan, V. Eidelman, and P. Resnik. 2010. cdec: A Decoder, Alignment, and Learning framework for finite-state and context-free translation models. In *Proceedings of ACL*.

K. Erk and S. Padó. 2008. A structured vector space model for word meaning in context. *Proceedings of EMNLP*.

J. R. Firth. 1957. A synopsis of linguistic theory 1930-55. 1952-59:1–32.

E. Grefenstette and M. Sadrzadeh. 2011. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of EMNLP*.

A. Haghighi, P. Liang, T. Berg-Kirkpatrick, and D. Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proceedings of ACL-HLT*.

K. M. Hermann and P. Blunsom. 2013. The Role of Syntax in Vector Space Models of Compositional Semantics. In *Proceedings of ACL*.

K. M. Hermann and P. Blunsom. 2014. Multilingual Distributed Representations without Word Alignment. In *Proceedings of ICLR*.

N. Kalchbrenner and P. Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality. *Proceedings of the ACL Workshop on Continuous Vector Space Models and their Compositionality*.

A. Klementiev, I. Titov, and B. Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *Proceedings of COLING*.

P. Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proceedings of the Machine Translation Summit*.

S. Lauly, A. Boulanger, and H. Larochelle. 2013. Learning multilingual word representations using a bag-of-words autoencoder. In *Deep Learning Workshop at NIPS*.

D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, December.

A. K. McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of INTERSPEECH*.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. *CoRR*.

T. Mikolov, Q. V. Le, and I. Sutskever. 2013b. Exploiting Similarities among Languages for Machine Translation. *CoRR*.

J. Mitchell and M. Lapata. 2008. Vector-based models of semantic composition. In *In Proceedings of ACL*.

A. Mnih and G. Hinton. 2009. A scalable hierarchical distributed language model. In *Proceedings of NIPS*.

J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng. 2011. Multimodal deep learning. In *ICML*.

D. Roy. 2003. Grounded spoken language acquisition: Experiments in word learning. *IEEE Transactions on Multimedia*, 5(2):197–209, June.

A. P. Sarath Chandar, M. K. Mitesh, B. Ravindran, V. Raykar, and A. Saha. 2013. Multilingual deep learning. In *Deep Learning Workshop at NIPS*.

R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of EMNLP*.

R. Socher, B. Huval, C. D. Manning, and A. Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP-CoNLL*, pages 1201–1211.

N. Srivastava and R. Salakhutdinov. 2012. Multimodal learning with deep boltzmann machines. In *Proceedings of NIPS*.

J. Turian, L. Ratinov, and Y. Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of ACL*.

P. D. Turney. 2012. Domain and function: A dual-space model of semantic relations and compositions. *Journal of Artificial Intelligence Research*, 44:533–585.

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

W. Y. Zou, R. Socher, D. Cer, and C. D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of EMNLP*.

Simple Negation Scope Resolution through Deep Parsing: A Semantic Solution to a Semantic Problem

Woodley Packard[♣], Emily M. Bender[♣], Jonathon Read[♠], Stephan Oepen[♡◇], and Rebecca Dridan[♡]

[♣] University of Washington, Department of Linguistics

[♠] Teesside University, School of Computing

[♡] University of Oslo, Department of Informatics

[◇] Potsdam University, Department of Linguistics

ebender@uw.edu, sweaglesw@sweaglesw.org, j.read@tees.ac.uk, {oe|rdridan}@ifi.uio.no

Abstract

In this work, we revisit Shared Task 1 from the 2012 *SEM Conference: the automated analysis of negation. Unlike the vast majority of participating systems in 2012, our approach works over explicit and formal representations of propositional semantics, i.e. derives the notion of negation *scope* assumed in this task from the structure of logical-form meaning representations. We relate the task-specific interpretation of (negation) scope to the concept of (quantifier and operator) scope in mainstream underspecified semantics. With reference to an explicit encoding of semantic predicate-argument structure, we can operationalize the annotation decisions made for the 2012 *SEM task, and demonstrate how a comparatively simple system for negation scope resolution can be built from an off-the-shelf deep parsing system. In a system combination setting, our approach improves over the best published results on this task to date.

1 Introduction

Recently, there has been increased community interest in the theoretical and practical analysis of what Morante and Sporleder (2012) call *modality and negation*, i.e. linguistic expressions that modulate the certainty or factuality of propositions. Automated analysis of such aspects of meaning is important for natural language processing tasks which need to consider the truth value of statements, such as for example text mining (Vincze et al., 2008) or sentiment analysis (Lapponi et al., 2012). Owing to its immediate utility in the curation of scholarly results, the analysis of negation and so-called hedges in bio-medical research literature has been the focus of several workshops, as well as the Shared Task at the 2011 Conference on Computational Language Learning (CoNLL).

Task 1 at the First Joint Conference on Lexical and Computational Semantics (*SEM 2012; Morante and Blanco, 2012) provided a fresh, principled annotation of negation and called for systems to analyze negation—detecting cues (affixes, words, or phrases that express negation), resolving their scopes (which parts of a sentence are actually negated), and identifying the negated event or property. The task organizers designed and documented an annotation scheme (Morante and Daelemans, 2012) and applied it to a little more than 100,000 tokens of running text by the novelist Sir Arthur Conan Doyle. While the task and annotations were framed from a semantic perspective, only one participating system actually employed explicit compositional semantics (Basile et al., 2012), with results ranking in the middle of the 12 participating systems. Conversely, the best-performing systems approached the task through machine learning or heuristic processing over *syntactic* and linguistically relatively *coarse-grained* representations; see § 2 below.

Example (1), where ⟨⟩ marks the cue and {} the in-scope elements, illustrates the annotations, including how negation inside a noun phrase can scope over discontinuous parts of the sentence.[1]

(1) {The German} was sent for but professed to {know} ⟨nothing⟩ {of the matter}.

In this work, we return to the 2012 *SEM task from a deliberately semantics-centered point of view, focusing on the hardest of the three sub-problems: scope resolution.[2] Where Morante and Daelemans (2012) characterize negation as an "extra-propositional aspect of meaning" (p. 1563),

[1] Our running example is a truncated variant of an item from the Shared Task training data. The remainder of the original sentence does not form part of the scope of this cue.

[2] Resolving negation scope is a more difficult sub-problem at least in part because (unlike cue and event identification) it is concerned with much larger, non-local and often discontinuous parts of each utterance. This intuition is confirmed by Read et al. (2012), who report results for each sub-problem using gold-standard inputs; in this setup, scope resolution showed by far the lowest performance levels.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 69–78,
Baltimore, Maryland, USA, June 23-25 2014.

we in fact see it as a core piece of compositionally constructed logical-form representations. Though the task-specific concept of scope of negation is not the same as the notion of quantifier and operator scope in mainstream underspecified semantics, we nonetheless find that reviewing the 2012 *SEM Shared Task annotations with reference to an explicit encoding of semantic predicate-argument structure suggests a simple and straightforward operationalization of their concept of negation scope. Our system implements these findings through a notion of functor-argument 'crawling', using as our starting point the underspecified logical-form meaning representations provided by a general-purpose deep parser.

Our contributions are three-fold: Theoretically, we correlate the structures at play in the Morante and Daelemans (2012) view on negation with formal semantic analyses; methodologically, we demonstrate how to approach the task in terms of underspecified, logical-form semantics; and practically, our combined system retroactively 'wins' the 2012 *SEM Shared Task. In the following sections, we review related work (§ 2), detail our own setup (§ 3), and present and discuss our experimental results (§ 4 and § 5, respectively).

2 Related Work

Read et al. (2012) describe the best-performing submission to Task 1 of the 2012 *SEM Conference. They investigated two approaches for scope resolution, both of which were based on syntactic constituents. Firstly, they created a set of 11 heuristics that describe the path from the preterminal of a cue to the constituent whose projection is predicted to match the scope. Secondly they trained an SVM ranker over candidate constituents, generated by following the path from a cue to the root of the tree and describing each candidate in terms of syntactic properties along the path and various surface features. Both approaches attempted to handle discontinuous instances by applying two heuristics to the predicted scope: (a) removing preceding conjuncts from the scope when the cue is in a conjoined phrase and (b) removing sentential adverbs from the scope. The ranking approach showed a modest advantage over the heuristics (with F_1 equal to 77.9 and 76.7, respectively, when resolving the scope of gold-standard cues in evaluation data). Read et al. (2012) noted however that the annotated scopes did not align with the Shared Task–provided constituents for 14% of the instances in the training data, giving an F_1 upper-bound of around 86.0 for systems that depend on those constituents.

Basile et al. (2012) present the only submission to Task 1 of the 2012 *SEM Conference which employed compositional semantics. Their scope resolution pipeline consisted primarily of the C&C parser and Boxer (Curran et al., 2007), which produce Discourse Representation Structures (DRSs). The DRSs represent negation explicitly, including representing other predications as being within the scope of negation. Basile et al. (2012) describe some amount of tailoring of the Boxer lexicon to include more of the Shared Task scope cues among those that produce the negation operator in the DRSs, but otherwise the system appears to directly take the notion of scope of negation from the DRS and project it out to the string, with one caveat: As with the logical-forms representations we use, the DRS logical forms do not include function words as predicates in the semantics. Since the Shared Task gold standard annotations included such arguably semantically vacuous (see Bender, 2013, p. 107) words in the scope, further heuristics are needed to repair the string-based annotations coming from the DRS-based system. Basile et al. resort to counting any words between in-scope tokens which are not themselves cues as in-scope. This simple heuristic raises their F_1 for full scopes from 20.1 to 53.3 on system-predicted cues.

3 System Description

The new system described here is what we call the MRS Crawler. This system operates over the normalized semantic representations provided by the LinGO English Resource Grammar (ERG; Flickinger, 2000).[3] The ERG maps surface strings to meaning representations in the format of Minimal Recursion Semantics (MRS; Copestake et al., 2005). MRS makes explicit predicate-argument relations, as well as partial information about scope (see below). We used the grammar together with one of its pre-packaged conditional Maximum Entropy models for parse ranking, trained on a combination of encyclopedia articles and tourism brochures. Thus, the deep parsing front-end system to our MRS Crawler has not been

[3] In our experiments, we use the 1212 release of the ERG, in combination with the ACE parser (`http://sweaglesw.org/linguistics/ace/`). The ERG and ACE are DELPH-IN resources; see `http://www.delph-in.net`.

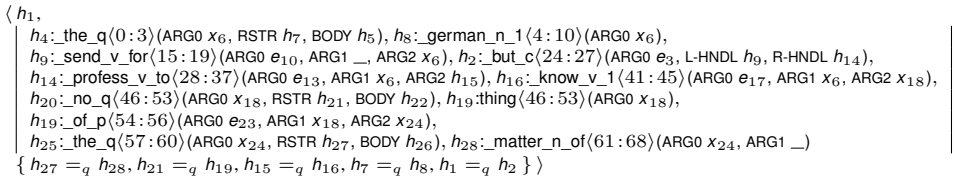


Figure 1: MRS analysis of our running example (1).

adapted to the task or its text type; it is applied in an 'off the shelf' setting. We combine our system with the outputs from the best-performing 2012 submission, the system of Read et al. (2012), firstly by relying on the latter for system negation cue detection,[4] and secondly as a fall-back in system combination as described in § 3.4 below.

Scopal information in MRS analyses delivered by the ERG fixes the scope of operators—such as negation, modals, scopal adverbs (including subordinating conjunctions like *while*), and clause-embedding verbs (e.g. *believe*)—based on their position in the constituent structure, while leaving the scope of quantifiers (e.g. *a* or *every*, but also other determiners) free. From these underspecified representations of possible scopal configurations, a scope resolution component can spell out the full range of fully-connected logical forms (Koller and Thater, 2005), but it turns out that such enumeration is not relevant here: the notion of scope encoded in the Shared Task annotations is not concerned with the relative scope of quantifiers and negation, such as the two possible readings of (2) represented informally below:[5]

(2) Everyone didn't leave.

a. $\forall(x)\neg\text{leave}(x) \sim$ Everyone stayed.

b. $\neg\forall(x)\text{leave}(x) \sim$ At least some stayed.

However, as shown below, the information about fixed scopal elements in an underspecified MRS is sufficient to model the Shared Task annotations.

3.1 MRS Crawling

Fig. 1 shows the ERG semantic analysis for our running example. The heart of the MRS is a multiset of elementary predications (EPs). Each elementary prediction includes a predicate symbol, a label (or 'handle', prefixed to predicates with a colon in Fig. 1), and one or more argument positions, whose values are semantic variables. Eventualities (e_i) in MRS denote states or activities, while instance variables (x_j) typically correspond to (referential or abstract) entities. All EPs have the argument position ARG0, called the *distinguished variable* (Oepen and Lønning, 2006), and no variable is the ARG0 of more than one non-quantifier EP.

The arguments of one EP are linked to the arguments of others either directly (sharing the same variable as their value), or indirectly (through so-called 'handle constraints', where $=_q$ in Fig. 1 denotes equality modulo quantifier insertion). Thus a well-formed MRS forms a connected graph. In addition, the grammar links the EPs to the elements of the surface string that give rise to them, via character offsets recorded in each EP (shown in angle brackets in Fig. 1). For the purposes of the present task, we take a negation cue as our entry point into the MRS graph (as our initial *active* EP), and then move through the graph according to the following simple operations to add EPs to the active set:

Argument Crawling Add to the scope all EPs whose distinguished variable or label is an argument of the active EP; for arguments of type h_k, treat any $=_q$ constraints as label equality.

Label Crawling Add all EPs whose label is identical to that of the active EP.

Functor Crawling Add all EPs that take the distinguished variable or label of the active EP as an argument (directly or via $=_q$ constraints).

Our MRS crawling algorithm is sketched in Fig. 2. To illustrate how the rules work, we will trace their operation in the analysis of example (1), i.e. traverse the EP graph in Fig. 1.

The negation cue is *nothing*, from character position 46 to 53. This leads us to _no_q as our entry point into the graph. Our algorithm states that for this type of cue (a quantifier) the first step is

[4] Read et al. (2012) predicted cues using a closed vocabulary assumption with a supervised classifier to disambiguate instances of cues.

[5] In other words, a possible semantic interpretation of the (string-based) Shared Task annotation guidelines and data is in terms of a quantifier-free approach to meaning representation, or in terms of one where quantifier scope need not be made explicit (as once suggested by, among others, Alshawi, 1992). From this interpretation, it follows that the notion of scope assumed in the Shared Task does not encompass interactions of negation operators and quantifiers.

1: Activate the cue EP
2: **if** the cue EP is a quantifier **then**
3: Activate EPs reached by functor crawling from the distinguished variable (ARG0) of the cue EP
4: **end if**
5: **repeat**
6: **for** each active EP X **do**
7: Activate EPs reached by argument crawling or label crawling unless they are co-modifiers of the negation cue.[a]
8: Activate EPs reached by functor crawling if they are modal verbs, or one of the following subordinating conjunctions reached by ARG1: *whether*, *when*, *because*, *to*, *with*, *although*, *unless*, *until*, or *as*.
9: **end for**
10: **until** a fixpoint is reached (no additional EPs were activated)
11: Deactivate zero-pronoun EPs (from imperative constructions)
12: Apply semantically empty word handling rules (iterate until a fixpoint is reached)
13: Apply punctuation heuristics

Figure 2: Algorithm for scope detection by MRS crawling

[a]Formally: If an EP shares its label with the negation cue, or is a quantifier whose restriction (RSTR) is $=_q$ equated with the label of the negation cue, it cannot be in-scope unless its ARG0 is an argument of the negation cue, or the ARG0 of the negation cue is one of its own arguments. See § 3.3 for elaboration.

functor crawling (see § 3.3 below), which brings _know_v_1 into the scope. We proceed with *argument crawling* and *label crawling*, which pick up _the_q⟨0:3⟩ and _german_n_1 as the ARG1. Further, as the ARG2 of _know_v_1, we reach thing and through recursive invocation we activate _of_p and, in yet another level of recursion, _the_q⟨57:60⟩ and _matter_n_of. At this point, crawling has no more links to follow. Thus, the MRS crawling operations 'paint' a subset of the MRS graph as in-scope for a given negation cue.

3.2 Semantically Empty Word Handling

Our crawling rules operate on semantic representations, but the annotations are with reference to the surface string. Accordingly, we need projection rules to map from the 'painted' MRS to the string. We can use the character offsets recorded in each EP to project the scope to the string. However, the string-based annotations also include words which the ERG treats as semantically vacuous. Thus in order to match the gold annotations, we define a set of heuristics for when to count vacuous words as in scope. In (1), there are no semantically empty words in-scope, so we illustrate these heuristics with another example:

(3) "I trust that {there is} ⟨nothing⟩ {of consequence which I have overlooked}?"

The MRS crawling operations discussed above paint the EPs corresponding to *is*, *thing*, *of*, *consequence*, *I*, and *overlooked* as in-scope (underlined in (3)). Conversely, the ERG treats the words *that*, *there*, *which*, and *have* as semantically empty. Of these, we need to add all except *that* to the scope. Our vacuous word handling rules use the syntactic structure provided by the ERG as scaffolding to help link the scope information gleaned from contentful words to vacuous words. Each node in the syntax tree is initially colored either in-scope or out-of-scope in agreement with the decision made by the crawler about the lexical head of the corresponding subtree. A semantically empty word is determined to be in-scope if there is an in-scope syntax tree node in the right position relative to it, as governed by a short list of templates organized by the type of the semantically empty word (particles, complementizers, non-referential pronouns, relative pronouns, and auxiliary verbs).

As an example, the rule for auxiliary verbs like *have* in our example (3) is that they are in scope when their verb phrase complement is in scope. Since *overlooked* is marked as in-scope by the crawler, the semantically empty *have* becomes in-scope as well. Sometimes the rules need to be iterated. For example, the main rule for relative pronouns is that they are in-scope when they fill a gap in an in-scope constituent; *which* fills a gap in the constituent *have overlooked*, but since *have* is the (syntactic) lexical head of that constituent, the verb phrase is not considered in-scope the first time the rules are tried.

Similar rules deal with *that* (complementizers are in-scope when the complement phrase is an argument of an in-scope verb, which is not the case here) and *there* (non-referential pronouns are in-scope when they are the subject of an in-scope VP, which is true here).

3.3 Re-Reading the Annotation Guidelines

Our MRS crawling algorithm was defined by looking at the annotated data rather than the annotation guidelines for the Shared Task (Morante et al., 2011). Nonetheless, our algorithm can be seen as a first pass formalization of the guidelines. In this section, we briefly sketch how our algorithm corresponds to different aspects of the guidelines.

For negated verbs, the guidelines state that "If the negated verb is the main verb in the sentence, the entire sentence is in scope." (Morante et al., 2011, 17). In terms of our operations defined over semantic representations, this is rendered as follows: all arguments of the negated verb are selected by *argument crawling*, all intersective modifiers by *label crawling*, and *functor crawling* (Fig. 2, line 8) captures modal auxiliaries and non-intersective modifiers. The guidelines treat predicative adjectives under a separate heading from verbs, but describe the same desired annotations (scope over the whole clause; ibid., p. 20). Since these structures are analogous in the semantic representations, the same operations that handle negated verbs also handle negated predicative adjectives correctly.

For negated subjects and objects, the guidelines state that the negation scopes over "all the clause" and "the clause headed by the verb" (Morante et al., 2011, 19), respectively. The examples given in the annotation guidelines suggest that these are in fact meant to refer to the same thing. The negation cue for a negated nominal argument will appear as a quantifier EP in the MRS, triggering line 3 of our algorithm. This *functor crawling* step will get to the verb's EP, and from there, the process is the same as the last two cases.

In contrast to subjects and objects, negation of a clausal argument is not treated as negation of the verb (ibid., p. 18). Since in this case, the negation cue will not be a quantifier in the MRS, there will be no *functor crawling* to the verb's EP.

For negated modifiers, the situation is somewhat more complex, and this is a case where our crawling algorithm, developed on the basis of the annotated data, does not align directly with the guidelines as given. The guidelines state that negated attributive adjectives have scope over the entire NP (including the determiner) (ibid., p. 20) and analogously negated adverbs have scope over the entire clause (ibid., p. 21). However, the annotations are not consistent, especially with respect to the treatment of negated adjectives: while the head noun and determiner (if present) are typically annotated as in scope, other co-modifiers, especially long, post-nominal modifiers (including relative clauses) are not necessarily included:

(4) "A dabbler in science, Mr. Holmes, a picker up of shells on the shores of {the} great ⟨un⟩{known ocean}.

(5) Our client looked down with a rueful face at {his} own ⟨un⟩{conventional appearance}.

(6) Here was {this} ⟨ir⟩{reproachable Englishman} ready to swear in any court of law that the accused was in the house all the time.

(7) {There is}, on the face of it, {something} ⟨un⟩{natural about this strange and sudden friendship between the young Spaniard and Scott Eccles}.

Furthermore, the guidelines treat relative clauses as subordinate clauses and thus negation inside a relative clause is treated as bound to that clause only, and includes neither the head noun of the relative clause nor any of its other dependents in its scope. However, from the perspective of MRS, a negated relative clause is indistinguishable from any other negated modifier of a noun. This treatment of relative clauses (as well as the inconsistencies in other forms of co-modification) is the reason for the exception noted at line 7 of Fig. 2. By disallowing the addition of EPs to the scope if they share the label of the negation cue but are not one of its arguments, we block the head noun's EP (and any EPs only reachable from it) in cases of relative clauses where the head verb inside the relative clause is negated. It also blocks co-modifiers like *great*, *own*, and the phrases headed by *ready* and *about* in (4)–(7). As illustrated in these examples, this is correct some but not all of the time. Having been unable to find a generalization capturing when comodifiers are annotated as in scope, we stuck with this approximation.

For negation within clausal modifiers of verbs, the annotation guidelines have further information, but again, our existing algorithm has the correct behavior: The guidelines state that a negation cue inside of the complement of a subordinating conjunction (e.g. *if*) has scope only over the subordinate clause (ibid., p. 18 and p. 26). The ERG treats all subordinating conjunctions as two-place predicates taking two scopal arguments. Thus, as with clausal complements of clause-embedding verbs, the embedding subordinating conjunction and any other arguments it might have are inaccessible, since functor crawling is restricted to a handful of specific configurations.

As is usually the case with exercises in formalization, our crawling algorithm generalizes beyond what is given explicitly in the annotation guidelines. For example, all arguments that are treated as semantically nominal (including PP arguments where the preposition is semantically null) are treated in the same way as subjects and objects; similarly, all arguments which are semantically clausal (including certain PP arguments) are handled the same way as clausal complements. This is possible because we take advantage of the high degree of normalization that the ERG accomplishes in mapping to the MRS representation.

There are also cases where we are more specific. The guidelines do not handle coordination in detail, except to state that in coordinated clauses negation is restricted to the clause it appears in (ibid., p. 17–18) and to include a few examples of coordination under the heading 'ellipsis'. In the case of VP coordination, our existing algorithm does not need any further elaboration to pick up the subject of the coordinated VP but not the non-negated conjunct, as shown in discussion of (1) in § 3.1 above. In the case of coordination of negated NPs, recall that to reach the main portion of the negated scope we must first apply functor crawling. The functor crawling procedure has a general mechanism to transparently continue crawling up through coordinated structures while blocking future crawling from traversing them again.[6]

On the other hand, there are some cases in the annotation guidelines which our algorithm does not yet handle. We have not yet provided any analysis of the special cases for *save* and *expect* discussed in Morante et al., 2011, pp. 22–23, and also do not have a means of picking out the overt verb in gapping constructions (p. 24).

Finally, we note that even carefully worked out annotation guidelines such as these are never followed perfectly consistently by the human annotators who apply them. Because our crawling algorithm so closely models the guidelines, this puts our system in an interesting position to provide feedback to the Shared Task organizers.

3.4 Fall-Back Configurations

The close match between our crawling algorithm and the annotation guidelines supported by the mapping to MRS provides for very high precision and recall when the analysis engine produces the desired MRS.[7] However, the analysis engine does not always provide the desired analysis, largely because of idiosyncrasies of the genre (e.g. vocatives appearing mid-sentence) that are either not handled by the grammar or not well modeled in the parse selection component. In addition, as noted above, there are a handful of negation cues we do not yet handle. Thus, we also tested fall-back configurations which use scope predictions based on MRS in some cases, and scope predictions from the system of Read et al. (2012) in others.

Our first fall-back configuration (Crawler$_N$ in Table 1) uses MRS-based predictions whenever there is a parse available and the cue is one that our system handles. Sometimes, the analysis picked by the ERG's statistical model is not the correct analysis for the given context. To combat such suboptimal parse selection performance, we investigated using the probability of the top ranked analysis (as determined by the parse selection model and conditioned on the sentence) as a confidence metric. Our second fall-back configuration (Crawler$_P$ in Table 1) uses MRS-based predictions when there is a parse available whose conditional probability is at least 0.5.[8]

4 Experiments

We evaluated the performance of our system using the Shared Task development and evaluation data (respectively CDD and CDE in Table 1). Since we do not attempt to perform cue detection, we report performance using gold cues and also using the system cues predicted by Read et al. (2012). We used the official Shared Task evaluation script to compute all scores.

4.1 Data Sets

The Shared Task data consists of chapters from the *Adventures of Sherlock Holmes* mystery novels and short stories. As such, the text is carefully edited turn-of-the-20th-century British English,[9]

[6] This allows *ate* to be reached in *We ate bread but no fish.*, while preventing *but* and *bread* from being reached, which they otherwise would via argument crawling from *ate*.

[7] And in fact, the task is somewhat noise-tolerant: some parse selection decisions are independent of each other, and a mistake in a part of the analysis far enough away from the negation cue does not harm performance.

[8] This threshold was determined empirically on the development data. We also experimented with other confidence metrics—the probability ratio of the top-ranked and second parse or the entropy over the probability distribution of the top 10 parses—but found no substantive differences.

[9] In contrast, the ERG was engineered for the analysis of contemporary American English, and an anecdotal analysis of parse failures and imperfect top-ranked parses suggests

		Gold Cues						System Cues					
		Scopes			Tokens			Scopes			Tokens		
Set	**Method**	**Prec**	**Rec**	**F_1**	**Prec**	**Rec**	**F_1**	**Prec**	**Rec**	**F_1**	**Prec**	**Rec**	**F_1**
CDD	Ranker	**100.0**	68.5	81.3	84.8	**86.8**	85.8	**91.7**	66.1	76.8	79.5	**84.9**	82.1
	Crawler	**100.0**	53.0	69.3	**89.3**	67.0	76.6	90.8	53.0	66.9	**84.7**	65.9	74.1
	Crawler$_N$	**100.0**	64.9	78.7	89.0	83.5	86.1	90.8	64.3	75.3	82.6	82.1	82.3
	Crawler$_P$	**100.0**	**70.2**	**82.5**	86.4	**86.8**	**86.6**	91.2	**67.9**	**77.8**	80.0	**84.9**	**82.4**
	Oracle	100.0	*76.8*	*86.9*	*91.5*	*89.1*	*90.3*						
CDE	Ranker	98.8	64.3	77.9	85.3	**90.7**	87.9	87.4	61.5	72.2	82.0	**88.8**	85.3
	Crawler	**100.0**	44.2	61.3	85.8	68.4	76.1	**87.8**	43.4	58.1	78.8	66.7	72.2
	Crawler$_N$	98.6	56.6	71.9	83.8	88.4	86.1	86.0	54.2	66.5	78.4	85.7	81.9
	Crawler$_P$	98.8	**65.5**	**78.7**	**86.1**	90.4	**88.2**	87.6	**62.7**	**73.1**	**82.6**	88.5	**85.4**
	Oracle	100.0	*70.3*	*82.6*	*89.5*	*93.1*	*91.3*						

Table 1: Scope resolution performance of various configurations over each subset of the Shared Task data. Ranker refers to the system of Read et al. (2012); Crawler refers to our current system in isolation, or falling back to the Ranker prediction either when the sentence is not covered by the parser (Crawler$_N$), or when the parse probability is predicted to be less than 0.5 (Crawler$_P$); finally, Oracle simulates best possible selection among the Ranker and Crawler predictions (and would be ill-defined on system cues).

annotated with token-level information about the cues and scopes in every negated sentence. The training set contains 848 negated sentences, the development set 144, and the evaluation set 235. As there can be multiple usages of negation in one sentence, this corresponds to 984, 173, and 264 instances, respectively.

Being rule-based, our system does not require any training data per se. However, the majority of our rule development and error analysis were performed against the designated training data. We used the designated development data for a single final round of error analysis and corrections. The system was declared frozen before running with the formal evaluation data. All numbers reported here reflect this frozen system.[10]

4.2 Results

Table 1 presents the results of our various configurations in terms of both (a) whole *scopes* (i.e. a true positive is only generated when the predicted scope matches the gold scope exactly) and (b) in-scope *tokens* (i.e. a true positive for every token the system correctly predicts to be in scope). The table also details the performance upper-bound for system combination, in which an oracle selects the system prediction which scores the greater token-wise F_1 for each gold cue.

The low recall levels for Crawler can be mostly attributed to imperfect parser coverage. Crawler$_N$, which falls back just for parse failure brings the recall back up, and results in F_1 levels closer to the system of Read et al. (2012), albeit still not quite advancing the state of the art (except over the development set). Our best results are from Crawler$_P$, which outperforms all other configurations on the development and evaluation sets.

The Oracle results are interesting because they show that there is much more to be gained in combining our semantics-based system with the Read et al. (2012) syntactically-focused system. Further analysis of these results to draw out the patterns of complementary errors and strengths is a promising avenue for future work.

4.3 Error Analysis

To shed more light on specific strengths and weaknesses of our approach, we performed a manual error analysis of scope predictions by Crawler, starting from gold cues so as to focus in-depth analysis on properties specific to scope resolution over MRSs. This analysis was performed on CDD, in order to not bar future work on this task. Of the 173 negation cue instances in CDD, Crawler by itself makes 94 scope predictions that exactly match the gold standard. In comparison, the system of Read et al. (2012) accomplishes 119 exact scope matches, of which 80 are shared with Crawler; in other words, there are 14 cue instances (or 8% of all cues) in which our approach can improve over the best-performing syntax-based submission to the original Shared Task.

that the archaic style in the 2012 *SEM Shared Task texts has a strong adverse effect on the parser.

[10]The code and data are available from `http://www.delph-in.net/crawler/`, for replicability (Fokkens et al., 2013).

We reviewed the 79 negation instances where Crawler made a wrong prediction in terms of exact scope match, categorizing the source of failure into five broad error types:

(1) Annotation Error In 11% of all instances, we consider the annotations erroneous or inconsistent. These judgments were made by two of the authors, who both were familiar with the annotation guidelines and conventions observable in the data. For example, Morante et al. (2011) unambiguously state that subordinating conjunctions shall not be in-scope (8), whereas relative pronouns should be (9), and a negated predicative argument to the copula must scope over the full clause (10):

(8) It was after nine this morning {when we} reached his house and {found} ⟨neither⟩ {you} ⟨nor⟩ {anyone else inside it}.

(9) "We can imagine that in the confusion of flight something precious, something which {he could} ⟨not⟩ {bear to part with}, had been left behind.

(10) He said little about the case, but from that little we gathered that he also was not ⟨dis⟩{satisfied} at the course of events.

(2) Parser Failure Close to 30% of Crawler failures reflect lacking coverage in the ERG parser, i.e. inputs for which the parser does not make available an analysis (within certain bounds on time and memory usage).[11] In this work, we have treated the ERG as an off-the-shelf system, but coverage could certainly be straightforwardly improved by adding analyses for phenomena particular to turn-of-the-20th-century British English.

(3) MRS Inadequacy Another 33% of our false scope predictions are Crawler-external, viz. owing to erroneous input MRSs due to imperfect disambiguation by the parser or other inadequacies in the parser output. Again, these judgments (assigning blame outside our own work) were double-checked by two authors, and we only counted MRS imperfections that actually involve the cue or in-scope elements. Here, we could anticipate improvements by training the parse ranker on in-domain data or otherwise adapting it to this task.

(4) Cue Selection In close to 9% of all cases, there is a valid MRS, but Crawler fails to pick out an initial EP that corresponds to the negation cue. This first type of genuine crawling failure often relates to cues expressed as affixation (11), as well as to rare usages of cue expressions that predominantly occur with different categories, e.g. *neither* as a generalized quantifier (12):

(11) Please arrange your thoughts and let me know, in their due sequence, exactly what those events are {which have sent you out} ⟨un⟩{brushed} and unkempt, with dress boots and waistcoat buttoned awry, in search of advice and assistance.

(12) You saw yourself {how} ⟨neither⟩ {of the inspectors dreamed of questioning his statement}, extraordinary as it was.

[11] Overall parsing coverage on this data is about 86%, but of course all parser failures on sentences containing negation surface in our error analysis of Crawler in isolation.

		Scopes			Tokens		
	Method	**Prec**	**Rec**	**F$_1$**	**Prec**	**Rec**	**F$_1$**
CDE	Boxer	76.1	41.0	53.3	69.2	82.3	75.2
	Crawler	87.8	43.4	58.1	78.8	66.7	72.2
	Crawler$_P$	87.6	62.7	73.1	82.6	88.5	85.4

Table 2: Comparison to Basile et al. (2012).

(5) Crawler Deficiency Finally, a little more than 16% of incorrect predictions we attribute to our crawling rules proper, where we see many instances of under-coverage of MRS elements (13, 14) and a few cases of extending the scope too wide (15). In the examples below, erroneous scope predictions by Crawler are indicated through underlining. Hardly any of the errors in this category, however, involve semantically vacuous tokens.

(13) He in turn had friends among the indoor servants who unite in {their} fear and ⟨dis⟩{like of their master}.

(14) He said little about the case, but from that little we gathered that {he also was} ⟨not⟩ {dissatisfied at the course of events}.

(15) I tell you, sir, {I could}n't move a finger, ⟨nor⟩ {get my breath}, till it whisked away and was gone.

5 Discussion and Comparison

The example in (1) nicely illustrates the strengths of the MRS Crawler and of the abstraction provided by the deep linguistic analysis made possible by the ERG. The negated verb in that sentence is *know*, and its first semantic argument is *The German*. This semantic dependency is directly and explicitly represented in the MRS, but the phrase expressing the dependent is not adjacent to the head in the string. Furthermore, even a system using syntactic structure to model scope would be faced with a more complicated task than our crawling rules: At the level of syntax the dependency is mediated by both verb phrase coordination and the control verb *profess*, as well as by the semantically empty infinitival marker *to*.

The system we propose is very similar in spirit to that of Basile et al. (2012). Both systems map from logical forms with explicit representations of scope of negation out to string-based annotations in the format provided by the Shared Task gold standard. The main points of difference are in the robustness of the system and in the degree of tailoring of both the rules for determining scope on the logical form level and the rules for handling semantically vacuous elements. The system description in Basile et al. (2012) suggests relatively little tailoring at either level: aside from adjustments to the Boxer lexicon to make more negation cues take the form of the negation operator in the DRS, the notion of scope is directly that given in the DRS. Similarly, their heuristic for picking up semantically vacuous words is string-based and straightforward. Our system, on the other hand, models the annotation guidelines more closely in the definition of the MRS crawling rules, and has more elaborated rules for handling semantically empty words. The Crawler alone is less robust than the Boxer-based system, returning no output for 29% of the cues in CDE. These factors all point to higher precision and lower recall for the Crawler compared to the Boxer-based system. At the token level, that is what we see. Since full-scope recall depends on token-level precision, the Crawler does better across the board at the full-scope level. A comparison of the results is shown in Table 2.

A final key difference between our results and those of Basile et al. (2012) is the cascading with a fall-back system. Presumably a similar system combination strategy could be pursued with the Boxer-based system in place of the Crawler.

6 Conclusion and Outlook

Our motivation in this work was to take the design of the 2012 *SEM Shared Task on negation analysis at face value—as an overtly *semantic* problem that takes a central role in our long-term pursuit of *language understanding*. Through both theoretical and practical reflection on the nature of representations at play in this task, we believe we have demonstrated that explicit semantic structure will be a key driver of further progress in the analysis of negation. We were able to closely align two independently developed semantic analyses—the negation-specific annotations of Morante et al. (2011), on the one hand, and the broad-coverage, MRS meaning representations of the ERG, on the other hand. In our view, the conceptual correlation between these two semantic views on negation analysis reinforces their credibility.

Unlike the rather complex top-performing systems from the original 2012 competition, our MRS Crawler is defined by a small set of general rules that operate over general-purpose, explicit meaning representations. Thus, our approach scores high on transparency, adaptability, and replicability. In isolation, the Crawler provides premium precision but comparatively low recall. Its limitations, we conjecture, reflect primarily on ERG parsing challenges and inconsistencies in the target data. In a sense, our approach pushes a larger proportion of the task into the parser, meaning (a) there should be good opportunities for parser adaptation to this somewhat idiosyncratic text type; (b) our results can serve to offer feedback on ERG semantic analyses and parse ranking; and (c) there is a much smaller proportion of very task-specific engineering. When embedded in a confidence-thresholded cascading architecture, our system advances the state of the art on this task, and oracle combination scores suggest there is much remaining room to better exploit the complementarity of approaches in our study. In future work, we will seek to better understand the division of labor between the systems involved through contrastive error analysis and possibly another oracle experiment, constructing gold-standard MRSs for part of the data. It would also be interesting to try a task-specific adaptation of the ERG parse ranking model, for example retraining on the pre-existing treebanks but giving preference to analyses that lead to correct Crawler results downstream.

Acknowledgments

We are grateful to Dan Flickinger, the main developer of the ERG, for many enlightening discussions and continuous assistance in working with the analyses available from the grammar. This work grew out of a discussion with colleagues of the Language Technology Group at the University of Oslo, notably Elisabeth Lien and Jan Tore Lønning, to whom we are indebted for stimulating cooperation. Furthermore, we have benefited from comments by participants of the 2013 DELPH-IN Summit, in particular Joshua Crowgey, Guy Emerson, Glenn Slayden, Sanghoun Song, and Rui Wang.

References

Alshawi, H. (Ed.). 1992. *The Core Language Engine*. Cambridge, MA, USA: MIT Press.

Basile, V., Bos, J., Evang, K., and Venhuizen, N. 2012. UGroningen. Negation detection with Discourse Representation Structures. In *Proceedings of the 1st Joint Conference on Lexical and Computational Semantics* (p. 301 – 309). Montréal, Canada.

Bender, E. M. 2013. *Linguistic fundamentals for natural language processing: 100 essentials from morphology and syntax*. San Rafael, CA, USA: Morgan & Claypool Publishers.

Copestake, A., Flickinger, D., Pollard, C., and Sag, I. A. 2005. Minimal Recursion Semantics. An introduction. *Research on Language and Computation*, *3*(4), 281 – 332.

Curran, J., Clark, S., and Bos, J. 2007. Linguistically motivated large-scale NLP with C&C and Boxer. In *Proceedings of the 45th Meeting of the Association for Computational Linguistics Demo and Poster Sessions* (p. 33 – 36). Prague, Czech Republic.

Flickinger, D. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, *6 (1)*, 15 – 28.

Fokkens, A., van Erp, M., Postma, M., Pedersen, T., Vossen, P., and Freire, N. 2013. Offspring from reproduction problems. What replication failure teaches us. In *Proceedings of the 51th Meeting of the Association for Computational Linguistics* (p. 1691 – 1701). Sofia, Bulgaria.

Koller, A., and Thater, S. 2005. Efficient solving and exploration of scope ambiguities. In *Proceedings of the 43rd Meeting of the Association for Computational Linguistics: Interactive Poster and Demonstration Sessions* (p. 9 – 12). Ann Arbor, MI, USA.

Lapponi, E., Read, J., and Øvrelid, L. 2012. Representing and resolving negation for sentiment analysis. In *Proceedings of the 2012 ICDM workshop on sentiment elicitation from natural text for information retrieval and extraction.* Brussels, Belgium.

Morante, R., and Blanco, E. 2012. *SEM 2012 Shared Task. Resolving the scope and focus of negation. In *Proceedings of the 1st Joint Conference on Lexical and Computational Semantics* (p. 265 – 274). Montréal, Canada.

Morante, R., and Daelemans, W. 2012. ConanDoyle-neg. Annotation of negation in Conan Doyle stories. In *Proceedings of the 8th International Conference on Language Resources and Evaluation.* Istanbul, Turkey.

Morante, R., Schrauwen, S., and Daelemans, W. 2011. *Annotation of negation cues and their scope guidelines v1.0* (Tech. Rep. # CTRS-003). Antwerp, Belgium: Computational Linguistics & Psycholinguistics Research Center, Universiteit Antwerpen.

Morante, R., and Sporleder, C. 2012. Modality and negation. An introduction to the special issue. *Computational Linguistics*, *38*(2), 223 – 260.

Oepen, S., and Lønning, J. T. 2006. Discriminant-based MRS banking. In *Proceedings of the 5th International Conference on Language Resources and Evaluation* (p. 1250 – 1255). Genoa, Italy.

Read, J., Velldal, E., Øvrelid, L., and Oepen, S. 2012. UiO1. Constituent-based discriminative ranking for negation resolution. In *Proceedings of the 1st Joint Conference on Lexical and Computational Semantics* (p. 310 – 318). Montréal, Canada.

Vincze, V., Szarvas, G., Farkas, R., Móra, G., and Csirik, J. 2008. The BioScope corpus. Biomedical texts annotated for uncertainty, negation and their scopes. *BMC Bioinformatics*, *9(Suppl 11)*.

Logical Inference on Dependency-based Compositional Semantics

Ran Tian **Yusuke Miyao** **Takuya Matsuzaki**
National Institute of Informatics, Japan
{tianran,yusuke,takuya-matsuzaki}@nii.ac.jp

Abstract

Dependency-based Compositional Semantics (DCS) is a framework of natural language semantics with easy-to-process structures as well as strict semantics. In this paper, we equip the DCS framework with logical inference, by defining *abstract denotations* as an abstraction of the computing process of denotations in original DCS. An inference engine is built to achieve inference on abstract denotations. Furthermore, we propose a way to generate on-the-fly knowledge in logical inference, by combining our framework with the idea of tree transformation. Experiments on FraCaS and PASCAL RTE datasets show promising results.

1 Introduction

Dependency-based Compositional Semantics (DCS) provides an intuitive way to model semantics of questions, by using simple dependency-like trees (Liang et al., 2011). It is expressive enough to represent complex natural language queries on a relational database, yet simple enough to be latently learned from question-answer pairs. In this paper, we equip DCS with *logical inference*, which, in one point of view, is "the best way of testing an NLP system's semantic capacity" (Cooper et al., 1996).

It should be noted that, however, a framework primarily designed for question answering is not readily suited for logical inference. Because, answers returned by a query depend on the specific database, but implication is independent of any databases. For example, answers to the question "*What books are read by students?*", should always be a subset of answers to "*What books are ever read by anyone?*", no matter how we store the data of students and how many records of books are there in our database.

Thus, our first step is to fix a notation which abstracts the calculation process of DCS trees, so as to clarify its meaning without the aid of any existing database. The idea is to borrow a minimal set of operators from relational algebra (Codd, 1970), which is already able to formulate the calculation in DCS and define *abstract denotation*, which is an abstraction of the *computation* of denotations guided by DCS trees. Meanings of sentences then can be represented by primary relations among abstract denotations. This formulation keeps the simpleness and computability of DCS trees mostly unaffected; for example, our semantic calculation for DCS trees is parallel to the denotation computation in original DCS.

An inference engine is built to handle inference on abstract denotations. Moreover, to compensate the lack of background knowledge in practical inference, we combine our framework with the idea of tree transformation (Bar-Haim et al., 2007), to propose a way of generating knowledge in logical representation from entailment rules (Szpektor et al., 2007), which are by now typically considered as syntactic rewriting rules.

We test our system on FraCaS (Cooper et al., 1996) and PASCAL RTE datasets (Dagan et al., 2006). The experiments show: (i) a competitive performance on FraCaS dataset; (ii) a big impact of our automatically generated on-the-fly knowledge in achieving high recall for a logic-based RTE system; and (iii) a result that outperforms state-of-the-art RTE system on RTE5 data. Our whole system is publicly released and can be downloaded from `http://kmcs.nii.ac.jp/tianran/tifmo/`.

2 The Idea

In this section we describe the idea of representing natural language semantics by DCS trees, and achieving inference by computing logical relations among the corresponding abstract denotations.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 79–89,
Baltimore, Maryland, USA, June 23-25 2014.

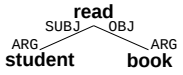


Figure 1: The DCS tree of "*students read books*"

student

ARG
Mark
John
Emily
...

book

ARG
A Tale of Two Cities
Ulysses
...

read

SUBJ	OBJ
Mark	New York Times
Mary	A Tale of Two Cities
John	Ulysses
...	...

Table 1: Databases of *student*, *book*, and *read*

2.1 DCS trees

DCS trees has been proposed to represent natural language semantics with a structure similar to dependency trees (Liang et al., 2011) (Figure 1). For the sentence "*students read books*", imagine a database consists of three tables, namely, a set of students, a set of books, and a set of "reading" events (Table 1). The DCS tree in Figure 1 is interpreted as a command for querying these tables, obtaining "reading" entries whose "SUBJ" field is **student** and whose "OBJ" field is **book**. The result is a set $\{$*John reads Ulysses*, ...$\}$, which is called a *denotation*.

DCS trees can be extended to represent linguistic phenomena such as quantification and coreference, with additional markers introducing additional operations on tables. Figure 2 shows an example with a quantifier "*every*", which is marked as "$\subset$" on the edge (**love**)OBJ−ARG(**dog**) and interpreted as a *division operator* $q_{\subset}^{\text{OBJ}}$ (§2.2). Optimistically, we believe DCS can provide a framework of semantic representation with sufficiently wide coverage for real-world texts.

The strict semantics of DCS trees brings us the idea of applying DCS to logical inference. This is not trivial, however, because DCS works under the assumption that databases are explicitly available. Obviously this is unrealistic for logical inference on unrestricted texts, because we cannot prepare a database for everything in the world. This fact fairly restricts the applicable tasks of DCS.

Our solution is to redefine DCS trees without the aid of any databases, by considering each node of a DCS tree as a content word in a sentence (but may no longer be a table in a specific database), while each edge represents semantic relations between two words. The labels on both ends of an edge, such as SUBJ (subject) and OBJ (object), are considered as *semantic roles* of the corresponding words[1]. To formulate the database querying process defined by a DCS tree, we provide formal semantics to DCS trees by employing *relational algebra* (Codd, 1970) for representing the query. As described below, we represent meanings of sentences with *abstract denotations*, and logical relations among sentences are computed as relations among their abstract denotations. In this way, we can perform inference over formulas of relational algebra, without computing database entries explicitly.

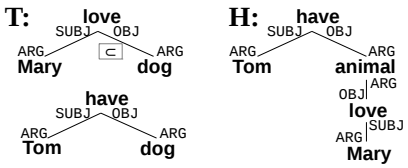


Figure 2: DCS trees of "*Mary loves every dog*" (Left-Up), "*Tom has a dog*" (Left-Down), and "*Tom has an animal that Mary loves*" (Right).

2.2 Abstract denotations

Abstract denotations are formulas constructed from a minimal set of relational algebra (Codd, 1970) operators, which is already able to formulate the database queries defined by DCS trees.

For example, the semantics of "*students read books*" is given by the abstract denotation:

$$F_1 = \textbf{read} \cap (\textbf{student}_{\text{SUBJ}} \times \textbf{book}_{\text{OBJ}}),$$

where **read**, **student** and **book** denote sets represented by these words respectively, and w_r represents the set w considered as the domain of the semantic role r (e.g. $\textbf{book}_{\text{OBJ}}$ is the set of books considered as objects). The operators $\cap$ and $\times$ represent intersection and Cartesian product respectively, both borrowed from relational algebra. It is not hard to see the abstract denotation denotes the intersection of the "*reading*" set (as illustrated by the "*read*" table in Table 1) with the product of "*student*" set and "*book*" set, which results in the same denotation as computed by the DCS tree in Figure 1, i.e. $\{$*John reads Ulysses*, ...$\}$. However, the point is that F_1 itself is an algebraic formula that does not depend on any concrete databases.

Formally, we introduce the following *constants*:

- W: a universal set containing all entities.

[1]The semantic role ARG is specifically defined for denoting nominal predicate.

	example phrase	abstract denotation / statement
compound noun	*pet fish*	**pet** $\cap$ **fish**
modification	*nice day*	**day** $\cap$ ($W_{\text{ARG}} \times$ **nice**$_{\text{MOD}}$)
temporal relation	*boys study at night*	**study** $\cap$ (**boy**$_{\text{SUBJ}} \times$ **night**$_{\text{TIME}}$)
relative clause	*books that students read*	**book** $\cap\, \pi_{\text{OBJ}}$(**read** $\cap$(**student**$_{\text{SUBJ}} \times W_{\text{OBJ}}$))
quantification	*all men die*	**man** $\subset\, \pi_{\text{SUBJ}}$(**die**)
hypernym		**dog** $\subset$ **animal**
derivation	*all criminals commit a crime*	**criminal** $\subset\, \pi_{\text{SUBJ}}$(**commit**$\cap$ ($W_{\text{SUBJ}} \times$ **crime**$_{\text{OBJ}}$))
antonym		**rise** $\|$ **fall**
negation	*no dogs are hurt*	**dog** $\|\, \pi_{\text{OBJ}}$(**hurt**)

Table 2: Abstract denotations and statements

- Content words: a content word (e.g. *read*) defines a set representing the word (e.g. **read** $= \{(x, y) \mid read(x, y)\}$).

In addition we introduce following *functions*:

- $\times$: the Cartesian product of two sets.
- $\cap$: the intersection of two sets.
- π_r: projection onto domain of semantic role r (e.g. $\pi_{\text{OBJ}}(\textbf{read}) = \{y \mid \exists x; read(x, y)\}$). Generally we admit projections onto multiple semantics roles, denoted by π_R where R is a set of semantic roles.
- ι_r: relabeling (e.g. $\iota_{\text{OBJ}}(\textbf{book}) = \textbf{book}_{\text{OBJ}}$).
- $q^r_{\subset}$: the division operator, where $q^r_{\subset}(A, B)$ is defined as the largest set X which satisfies $B_r \times X \subset A$.[2] This is used to formulate universal quantifiers, such as "*Mary loves every dog*" and "*books read by all students*".

An *abstract denotation* is then defined as finite applications of functions on either constants or other abstract denotations.

2.3 Statements

As the semantics of DCS trees is formulated by abstract denotations, the meanings of declarative sentences are represented by *statements* on abstract denotations. Statements are declarations of some relations among abstract denotations, for which we consider the following set relations:

Non-emptiness $A \neq \emptyset$: the set A is not empty.
Subsumption $A \subset B$: set A is subsumed by B.[3]

Roughly speaking, the relations correspond to the logical concepts *satisfiability* and *entailment*.

[2] If A and B has the same dimension, $q_{\subset}(A, B)$ is either $\emptyset$ or $\{*\}$ (0-dimension point set), depending on if $A \subset B$.

[3] Using division operator, subsumption can be represented by non-emptiness, since for sets A, B of the same dimension, $q_{\subset}(A, B) \neq \emptyset \Leftrightarrow A \subset B$.

Abstract denotations and statements are convenient for representing semantics of various types of expressions and linguistic knowledge. Some examples are shown in Table 2.[4]

2.4 Logical inference on DCS

Based on abstract denotations, we briefly describe our process to apply DCS to textual inference.

2.4.1 Natural language to DCS trees

To obtain DCS trees from natural language, we use Stanford CoreNLP[5] for dependency parsing (Socher et al., 2013), and convert Stanford dependencies to DCS trees by pattern matching on POS tags and dependency labels.[6] Currently we use the following semantic roles: `ARG`, `SUBJ`, `OBJ`, `IOBJ`, `TIME` and `MOD`. The semantic role `MOD` is used for any restrictive modifiers. Determiners such as "all", "every" and "each" trigger quantifiers, as shown in Figure 2.

2.4.2 DCS trees to statements

A DCS tree $\mathcal{T} = (\mathcal{N}, \mathcal{E})$ is defined as a rooted tree, where each node $\sigma \in \mathcal{N}$ is labeled with a content word $w(\sigma)$ and each edge $(\sigma, \sigma') \in \mathcal{E} \subset \mathcal{N} \times \mathcal{N}$ is labeled with a pair of semantic roles (r, r')[7]. Here σ is the node nearer to the root. Furthermore, for each edge (σ, σ') we can optionally assign a quantification marker.

Abstract denotation of a DCS tree can be calculated in a bottom-up manner. For example, the abstract denotation of **H** in Figure 2 is calculated from the leaf node **Mary**, and then:

Node **love** (*Mary loves*):
$F_2 = \textbf{love} \cap (\textbf{Mary}_{\text{SUBJ}} \times W_{\text{OBJ}})$
Node **animal** (*Animal that Mary loves*):
$F_3 = \textbf{animal} \cap \pi_{\text{OBJ}}(F_2)$
Node **have** (*Tom has an animal that Mary loves*):
$F_4 = \textbf{have} \cap (\textbf{Tom}_{\text{SUBJ}} \times (F_3)_{\text{OBJ}})$.

Formally, suppose the root σ of a DCS tree $\mathcal{T}$ has children $\tau_1, \ldots, \tau_n$, and edges $(\sigma, \tau_1), \ldots, (\sigma, \tau_n)$ labeled by $(r_1, r'_1), \ldots, (r_n, r'_n)$, respectively. The abstract denotation of $\mathcal{T}$ is defined as:

$$[\![\mathcal{T}]\!] = w(\sigma) \cap (\bigcap_{i=1}^{n} \iota_{r_i}(\pi_{r'_i}([\![\mathcal{T}_{\tau_i}]\!])) \times W_{R_\sigma \setminus r_i}),$$

[4] Negation and disjointness ("$\|$") are explained in §2.5.

[5] `http://nlp.stanford.edu/software/corenlp.shtml`

[6] In (Liang et al., 2011) DCS trees are learned from QA pairs and database entries. We obtain DCS trees from dependency trees, to bypass the need of a concrete database.

[7] The definition differs slightly from the original Liang et al. (2011), mainly for the sake of simplicity and clarity.

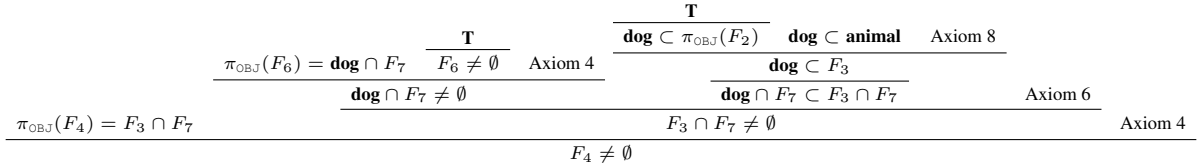


Figure 3: An example of proof using abstract denotations

1. $W \neq \emptyset$	5. $(A \subset B \;\&\; B \subset C) \Rightarrow A \subset C$
2. $A \cap B \subset A$	6. $(A \subset B \;\&\; A \neq \emptyset) \Rightarrow B \neq \emptyset$
3. $B_r \times q^r_{\subset}(A, B) \subset A$	7. $A \subset B \Rightarrow \pi_R(A) \subset \pi_R(B)$
4. $\pi_R(A) \neq \emptyset \Leftrightarrow A \neq \emptyset$	8. $(C \subset A \;\&\; C \subset B) \Rightarrow C \subset A \cap B$

Table 3: An excerpt of axioms

where $\mathcal{T}_{\tau_i}$ is the subtree of $\mathcal{T}$ rooted at τ_i, and R_σ is the set of possible semantic roles for content word $w(\sigma)$ (e.g. $R_{\textbf{love}} = \{\texttt{SUBJ}, \texttt{OBJ}\}$), and $W_{R_\sigma \setminus r_i}$ is the product of W which has dimension $R_\sigma \setminus r_i$ (e.g. $W_{\{\texttt{SUBJ},\texttt{OBJ}\}\setminus\texttt{SUBJ}} = W_{\texttt{OBJ}}$).

When universal quantifiers are involved, we need to add division operators to the formula. If (σ, τ_i) is assigned by a quantification marker "$\subset$"[8], then the abstract denotation is[9]

$$[\![\mathcal{T}]\!] = q^{r_i}_{\subset}(\pi_{R_\sigma \setminus \{r_1,\ldots,r_{i-1}\}}([\![\mathcal{T}']\!]), \pi_{r'_i}([\![\mathcal{T}_{\tau_i}]\!])),$$

where $\mathcal{T}'$ is the same tree as $\mathcal{T}$ except that the edge (σ, τ_i) is removed. For example, the abstract denotation of the first sentence of **T** in Figure 2 (*Mary loves every dog*) is calculated from F_2 (*Mary loves*) as

$$F_5 = q^{\texttt{OBJ}}_{\subset}(\pi_{\texttt{OBJ}}(F_2), \textbf{dog}).$$

After the abstract denotation $[\![\mathcal{T}]\!]$ is calculated, the statement representing the meaning of the sentence is defined as $[\![\mathcal{T}]\!] \neq \emptyset$. For example, the statement of "*students read books*" is $\textbf{read} \cap (\textbf{student}_{\texttt{SUBJ}} \times \textbf{book}_{\texttt{OBJ}}) \neq \emptyset$, and the statement of "*Mary loves every dog*" is $q^{\texttt{OBJ}}_{\subset}(\pi_{\texttt{OBJ}}(F_2), \textbf{dog}) \neq \emptyset$, which is logically equivalent to $\textbf{dog} \subset \pi_{\texttt{OBJ}}(F_2)$.[10]

2.4.3 Logical inference

Since meanings of sentences are represented by statements on abstract denotations, logical inference among sentences is reduced to deriving new relations among abstract denotations. This is done by applying axioms to known statements, and approximately 30 axioms are implemented (Table 3).

[8]Multiple quantifiers can be processed similarly.

[9]The result of $[\![\mathcal{T}]\!]$ depends on the order of the children $\tau_1, \ldots, \tau_n$. Different orders correspond to readings of different quantifier scopes.

[10]See Footnote 2,3.

These are algebraic properties of abstract denotations, among which we choose a set of axioms that can be handled efficiently and enable most common types of inference seen in natural language.

For the example in Figure 2, by constructing the following abstract denotations:

Tom has a dog:

$$F_6 = \textbf{have} \cap (\textbf{Tom}_{\texttt{SUBJ}} \times \textbf{dog}_{\texttt{OBJ}})$$

Objects *that Tom has*:

$$F_7 = \pi_{\texttt{OBJ}}(\textbf{have} \cap (\textbf{Tom}_{\texttt{SUBJ}} \times W_{\texttt{OBJ}})),$$

we can use the lexical knowledge $\textbf{dog} \subset \textbf{animal}$, the statements of **T** (i.e. $\textbf{dog} \subset \pi_{\texttt{OBJ}}(F_2)$ and $F_6 \neq \emptyset$), and the axioms in Table 3,[11] to prove the statement of **H** (i.e. $F_4 \neq \emptyset$) (Figure 3).

We built an inference engine to perform logical inference on abstract denotations as above. In this logical system, we treat abstract denotations as *terms* and statements as *atomic sentences*, which are far more easier to handle than first order predicate logic (FOL) formulas. Furthermore, all implemented axioms are horn clauses, hence we can employ forward-chaining, which is very efficient.

2.5 Extensions

Further extensions of our framework are made to deal with additional linguistic phenomena, as briefly explained below.

Negation To deal with negation in our forward-chaining inference engine, we introduce one more relation on abstract denotations, namely *disjointness* $A \parallel B$, meaning that A and B are disjoint sets. Using disjointness we implemented two types of negations: (i) atomic negation, for each content word w we allow negation $\bar{w}$ of that word, characterized by the property $w \parallel \bar{w}$; and (ii) root negation, for a DCS tree $\mathcal{T}$ and its denotation $[\![\mathcal{T}]\!]$, the negation of $\mathcal{T}$ is represented by $\mathcal{T} \parallel \mathcal{T}$, meaning that $\mathcal{T} = \emptyset$ in its effect.

Selection Selection operators in relational algebra select a subset from a set to satisfy some spe-

[11]Algebraic identities, such as $\pi_{\texttt{OBJ}}(F_4) = F_3 \cap F_7$ and $\pi_{\texttt{OBJ}}(F_6) = \textbf{dog} \cap F_7$, are also axioms.

cific properties. This can be employed to represent linguistic phenomena such as downward monotonicity and generalized quantifiers. In the current system, we implement (i) superlatives, e.g. $s_{highest}(\textbf{mountain} \cap (W_{\texttt{ARG}} \times \textbf{Asia}_{\texttt{MOD}}))$ (the highest mountain in Asia) and (ii) numerics, e.g. $s_{two}(\textbf{pet} \cap \textbf{fish})$ (two pet fish), where s_f is a selection marker. Selection operators are implemented as markers assigned to abstract denotations, with specially designed axioms. For example superlatives satisfy the following property: $A \subset B \ \& \ s_{highest}(B) \subset A \Rightarrow s_{highest}(B) = s_{highest}(A)$. New rules can be added if necessary.

Coreference We use Stanford CoreNLP to resolve coreferences (Raghunathan et al., 2010), whereas coreference is implemented as a special type of selection. If a node σ in a DCS tree $\mathcal{T}$ belongs to a mention cluster m, we take the abstract denotation $[\![\mathcal{T}_\sigma]\!]$ and make a selection $s_m([\![\mathcal{T}_\sigma]\!])$, which is regarded as the abstract denotation of that mention. Then all selections of the same mention cluster are declared to be equal.

3 Generating On-the-fly Knowledge

Recognizing textual entailment (RTE) is the task of determining whether a given textual statement **H** can be inferred by a text passage **T**. For this, our primary textual inference system operates as:

1. For a **T**-**H** pair, apply dependency parsing and coreference resolution.
2. Perform rule-based conversion from dependency parses to DCS trees, which are translated to statements on abstract denotations.
3. Use statements of **T** and linguistic knowledge as premises, and try to prove statements of **H** by our inference engine.

However, this method does not work for real-world datasets such as PASCAL RTE (Dagan et al., 2006), because of the knowledge bottleneck: it is often the case that the lack of sufficient linguistic knowledge causes failure of inference, thus the system outputs "no entailment" for almost all pairs (Bos and Markert, 2005).

The transparent syntax-to-semantics interface of DCS enables us to back off to NLP techniques during inference for catching up the lack of knowledge. We extract fragments of DCS trees as paraphrase candidates, translate them back to linguistic expressions, and apply distributional similarity to judge their validity. In this way, our framework combines distributional and logical semantics, which is also the main subject of Lewis and Steedman (2013) and Beltagy et al. (2013).

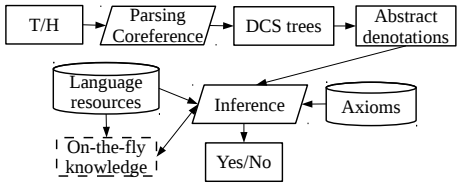


Figure 4: RTE system

As follows, our full system (Figure 4) additionally invokes linguistic knowledge on-the-fly:

4. If **H** is not proven, compare DCS trees of **T** and **H**, and generate path alignments.
5. Aligned paths are evaluated by a similarity score to estimate their likelihood of being paraphrases. Path alignments with scores higher than a threshold are accepted.
6. Convert accepted path alignments into statements on abstract denotations, use them in logical inference as new knowledge, and try to prove **H** again.

3.1 Generating path alignments

On-the-fly knowledge is generated by aligning paths in DCS trees. A path is considered as joining two *germs* in a DCS tree, where a *germ* is defined as a specific semantic role of a node. For example, Figure 5 shows DCS trees of the following sentences (a simplified pair from RTE2-dev):

T: *Tropical storm Debby is blamed for deaths.*
H: *A storm has caused loss of life.*

The germ $\texttt{OBJ}(\textbf{blame})$ and germ $\texttt{ARG}(\textbf{death})$ in DCS tree of **T** are joined by the underscored path. Two paths are aligned if the joined germs are aligned, and we impose constraints on aligned germs to inhibit meaningless alignments, as described below.

3.2 Aligning germs by logical clues

Two germs are aligned if they are both at leaf nodes (e.g. $\texttt{ARG}(\textbf{death})$ in **T** and $\texttt{ARG}(\textbf{life})$ in **H**, Figure 5), or they already have part of their meanings in common, by some logical clues.

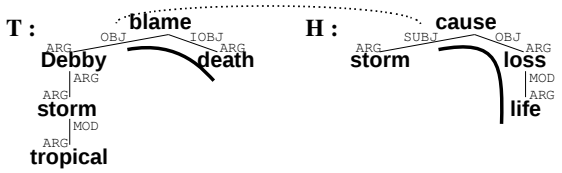


Figure 5: Aligned paths (underscored by the solid lines) and aligned germs (joined by the dotted line)

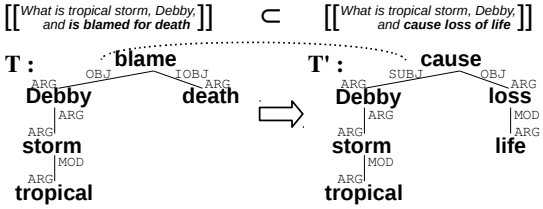


Figure 6: Tree transformation and generated on-the-fly knowledge (subsumption of denotations shown above the trees)

To formulate this properly, we define the abstract denotation of a germ, which, intuitively, represents the meaning of the germ in the specific sentence. The abstract denotation of a germ is defined in a top-down manner: for the root node ρ of a DCS tree $\mathcal{T}$, we define its denotation $[\![\rho]\!]_{\mathcal{T}}$ as the denotation of the entire tree $[\![\mathcal{T}]\!]$; for a non-root node τ and its parent node σ, let the edge (σ, τ) be labeled by semantic roles (r, r'), then define

$$[\![\tau]\!]_{\mathcal{T}} = [\![\mathcal{T}_\tau]\!] \cap (\iota_{r'}(\pi_r([\![\sigma]\!]_{\mathcal{T}})) \times W_{R_\tau \setminus r'}).$$

Now for a germ $r(\sigma)$, the denotation is defined as the projection of the denotation of node σ onto the specific semantic role r: $[\![r(\sigma)]\!]_{\mathcal{T}} = \pi_r([\![\sigma]\!]_{\mathcal{T}})$.

For example, the abstract denotation of germ ARG(**book**) in Figure 1 is defined as $\pi_{\texttt{ARG}}(\textbf{book} \cap \pi_{\texttt{OBJ}}(\textbf{read} \cap (\textbf{student}_{\texttt{SUBJ}} \times \textbf{book}_{\texttt{OBJ}})))$, meaning "*books read by students*". Similarly, denotation of germ OBJ(**blame**) in **T** of Figure 5 indicates the object of "*blame*" as in the sentence "*Tropical storm Debby is blamed for death*", which is a *tropical storm*, is *Debby*, etc. Technically, each germ in a DCS tree indicates a variable when the DCS tree is translated to a FOL formula, and the abstract denotation of the germ corresponds to the set of *consistent values* (Liang et al., 2011) of that variable.

The logical clue to align germs is: if there exists an abstract denotation, other than W, that is a superset of both abstract denotations of two germs, then the two germs can be aligned. A simple example is that ARG(**storm**) in **T** can be aligned to ARG(**storm**) in **H**, because their denotations have a common superset other than W, namely $\pi_{\texttt{ARG}}(\textbf{storm})$. A more complicated example is that OBJ(**blame**) and SUBJ(**cause**) can be aligned, because inference can induce $[\![\texttt{OBJ}(\textbf{blame})]\!]_{\mathbf{T}} = [\![\texttt{ARG}(\textbf{Debby})]\!]_{\mathbf{T}} = [\![\texttt{ARG}(\textbf{storm})]\!]_{\mathbf{T}}$, as well as $[\![\texttt{SUBJ}(\textbf{cause})]\!]_{\mathbf{H}} = [\![\texttt{ARG}(\textbf{storm})]\!]_{\mathbf{H}}$, so they also have the common superset $\pi_{\texttt{ARG}}(\textbf{storm})$. However, for example, logical clues can avoid aligning ARG(**storm**) to ARG(**loss**), which is obviously meaningless.

3.3 Scoring path alignments by similarity

Aligned paths are evaluated by a similarity score, for which we use distributional similarity of the words that appear in the paths (§4.1). Only path alignments with high similarity scores can be accepted. Also, we only accept paths of length ≤ 5, to prevent too long paths to be aligned.

3.4 Applying path alignments

Accepted aligned paths are converted into statements, which are used as new knowledge. The conversion is done by first performing a DCS tree transformation according to the aligned paths, and then declare a subsumption relation between the denotations of aligned germs. For example, to apply the aligned path pair generated in Figure 5, we use it to transform **T** into a new tree **T'** (Figure 6), and then the aligned germs, OBJ(**blame**) in **T** and SUBJ(**cause**) in **T'**, will generate the on-the-fly knowledge: $[\![\texttt{OBJ}(\textbf{blame})]\!]_{\mathbf{T}} \subset [\![\texttt{SUBJ}(\textbf{cause})]\!]_{\mathbf{T'}}$.

Similar to the tree transformation based approach to RTE (Bar-Haim et al., 2007), this process can also utilize lexical-syntactic entailment rules (Szpektor et al., 2007). Furthermore, since the on-the-fly knowledge is generated by transformed pairs of DCS trees, all contexts are preserved: in Figure 6, though the tree transformation can be seen as generated from the entailment rule "*X is blamed for death* → *X causes loss of life*", the generated on-the-fly knowledge, as shown above the trees, only fires with the additional condition that X is a *tropical storm* and is *Debby*. Hence, the process can also be used to generate knowledge from context sensitive rules (Melamud et al., 2013), which are known to have higher quality (Pantel et al., 2007; Clark and Harrison, 2009).

However, it should be noted that using on-the-fly knowledge in logical inference is not a trivial

task. For example, the FOL formula of the rule "*X is blamed for death → X causes loss of life*" is:

$$\forall x; (\exists a; blame(x, a) \ \& \ death(a)) \rightarrow \\ (\exists b, c; cause(x, b) \ \& \ loss(b, c) \ \& \ life(c)),$$

which is not a horn clause. The FOL formula for the context-preserved rule in Figure 6 is even more involved. Still, it can be efficiently treated by our inference engine because as a statement, the formula $[\![\texttt{OBJ}(\textbf{blame})]\!]_{\mathbf{T}} \subset [\![\texttt{SUBJ}(\textbf{cause})]\!]_{\mathbf{T}'}$ is an atomic sentence, more than a horn clause.

4 Experiments

In this section, we evaluate our system on FraCaS (§4.2) and PASCAL RTE datasets (§4.3).

4.1 Language Resources

The lexical knowledge we use are synonyms, hypernyms and antonyms extracted from WordNet[12]. We also add axioms on named entities, stopwords, numerics and superlatives. For example, named entities are singletons, so we add axioms such as $\forall x; (x \subset \textbf{Tom} \ \& \ x \neq \emptyset) \rightarrow \textbf{Tom} \subset x$.

To calculate the similarity scores of path alignments, we use the sum of word vectors of the words from each path, and calculate the cosine similarity. For example, the similarity score of the path alignment "OBJ(**blame**)IOBJ-ARG(**death**) ≈ SUBJ(**cause**)OBJ-ARG(**loss**)MOD-ARG(**life**)" is calculated as the cosine similarity of vectors **blame+death** and **cause+loss+life**. Other structures in the paths, such as semantic roles, are ignored in the calculation. The word vectors we use are from Mikolov et al. (2013)[13] (*Mikolov13*), and additional results are also shown using Turian et al. (2010)[14] (*Turian10*). The threshold for accepted path alignments is set to 0.4, based on preexperiments on RTE development sets.

4.2 Experiments on FraCaS

The FraCaS test suite contains 346 inference problems divided into 9 sections, each focused on a category of semantic phenomena. We use the data by MacCartney and Manning (2007), and experiment on the first section, *Quantifiers*, following Lewis and Steedman (2013). This section has 44 single premise and 30 multi premise problems. Most of the problems do not require lexical knowledge, so we use our primary textual inference system without on-the-fly knowledge nor WordNet, to test the performance of the DCS framework as formal semantics. To obtain the three-valued output (i.e. *yes*, *no*, and *unknown*), we output "*yes*" if **H** is proven, or try to prove the negation of **H** if **H** is not proven. To negate **H**, we use the root negation as described in §2.5. If the negation of **H** is proven, we output "*no*", otherwise we output "*unknown*".

	Single Prem.	Multi Prem.
Lewis13	70	50
MacCartney07	84.1	-
MacCartney08	**97.7**	-
Our Sys.	79.5	**80.0**

Table 4: Accuracy (%) on FraCaS

The result is shown in Table 4. Since our system uses an off-the-shelf dependency parser, and semantic representations are obtained from simple rule-based conversion from dependency trees, there will be only one (right or wrong) interpretation in face of ambiguous sentences. Still, our system outperforms Lewis and Steedman (2013)'s probabilistic CCG-parser. Compared to MacCartney and Manning (2007) and MacCartney and Manning (2008), our system does not need a pretrained alignment model, and it improves by making multi-sentence inferences. To sum up, the result shows that DCS is good at handling universal quantifiers and negations.

Most errors are due to wrongly generated DCS trees (e.g. wrongly assigned semantic roles) or unimplemented quantifier triggers (e.g. "*neither*") or generalized quantifiers (e.g. "*at least a few*"). These could be addressed by future work.

4.3 Experiments on PASCAL RTE datasets

On PASCAL RTE datasets, strict logical inference is known to have very low recall (Bos and Markert, 2005), so on-the-fly knowledge is crucial in this setting. We test the effect of on-the-fly knowledge on RTE2, RTE3, RTE4 and RTE5 datasets, and compare our system with other approaches.

4.3.1 Impact of on-the-fly knowledge

Results on test data are shown in Table 5. When only primary knowledge is used in inference (the first row), recalls are actually very low; After we activate the on-the-fly knowledge, recalls jump to over 50%, with a moderate fall of precision. As a result, accuracies significantly increase.

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

[13] http://code.google.com/p/word2vec/

[14] http://metaoptimize.com/projects/wordreprs/

	RTE2			RTE3			RTE4			RTE5		
	Prec.	Rec.	Acc.	Prec.	Rec.	Acc.	Prec.	Rec.	Acc.	Prec.	Rec.	Acc.
Primary	**70.9**	9.8	52.9	**73.2**	7.3	51.1	**89.7**	5.2	52.3	**82.6**	6.3	52.5
+On-the-fly	57.6	**66.5**	**58.8**	63.7	**64.6**	**63.0**	60.0	**57.4**	**59.6**	69.9	**55.7**	**65.8**

Table 5: Impact of on-the-fly knowledge

	RTE2	RTE3	RTE4	RTE5
Bos06	60.6	-	-	-
MacCartney08	-	59.4	-	-
Clark08	-	-	56.5	-
Wang10	**63.0**	61.1	-	-
Stern11	61.6	**67.1**	-	63.5
Stern12	-	-	-	64.0
Our Sys.	58.8	63.0	**59.6**	**65.8**

Table 6: Comparison with other systems

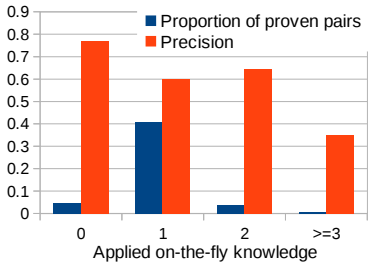


Figure 7: Proportion of proven pairs and their precision, w.r.t. pieces of on-the-fly knowledge.

4.3.2 Comparison to other RTE systems

A comparison between our system and other RTE systems is shown in Table 6. Bos06 (Bos and Markert, 2006) is a hybrid system combining deep features from a theorem prover and a model builder, together with shallow features such as lexical overlap and text length. MacCartney08 (MacCartney and Manning, 2008) uses natural logic to calculate inference relations between two superficially aligned sentences. Clark08 (Clark and Harrison, 2008) is a logic-based system utilizing various resources including WordNet and DIRT paraphrases (Lin and Pantel, 2001), and is tolerant to partially unproven **H** sentences in some degree. All of the three systems pursue a logical approach, while combining various techniques to achieve robustness. The result shows that our system has comparable performance. On the other hand, Wang10 (Wang and Manning, 2010) learns a tree-edit model from training data, and captures entailment relation by tree edit distance. Stern11 (Stern and Dagan, 2011) and Stern12 (Stern et al., 2012) extend this framework to utilize entailment rules as tree transformations. These are more tailored systems using machine learning with many handcrafted features. Still, our unsupervised system outperforms the state-of-the-art on RTE5 dataset.

4.3.3 Analysis

Summing up test data from RTE2 to RTE5, Figure 7 shows the proportion of all proven pairs and their precision. Less than 5% pairs can be proven primarily, with a precision of 77%. Over 40% pairs can be proven by one piece of on-the-fly knowledge, yet pairs do exist in which more than 2 pieces are necessary. The precisions of 1 and 2 pieces on-the-fly knowledge application are over 60%, which is fairly high, given our rough estimation of the similarity score. As a comparison, Dinu and Wang (2009) studied the proportion of proven pairs and precision by applying DIRT rules to tree skeletons in RTE2 and RTE3 data. The proportion is 8% with precision 65% on RTE2, and proportion 6% with precision 72% on RTE3. Applied by our logical system, the noisy on-the-fly knowledge can achieve a precision comparable to higher quality resources such as DIRT.

A major type of error is caused by the ignorance of semantic roles in calculation of similarity scores. For example, though "*Italy beats Kazakhstan*" is not primarily proven from "*Italy is defeated by Kazakhstan*", our system does produce the path alignment "SUBJ(**beat**)OBJ $\approx$ OBJ(**defeat**)SUBJ" with a high similarity score. The impact of such errors depends on the data making methodology, though. It lowers precisions in RTE2 and RTE3 data, particularly in "IE" subtask (where precisions drop under 0.5). On the other hand, it occurs less often in "IR" subtask.

Finally, to see if we "get lucky" on RTE5 data in the choice of word vectors and thresholds, we change the thresholds from 0.1 to 0.7 and draw the precision-recall curve, using two types of word vectors, *Mikolov13* and *Turian10*. As shown in Figure 8, though the precision drops for *Turian10*, both curves show the pattern that our system keeps gaining recall while maintaining precision to a certain level. Not too much "magic" in *Mikolov13* actually: for over 80% pairs, every node in DCS tree of **H** can be covered by a path of length ≤ 5 that

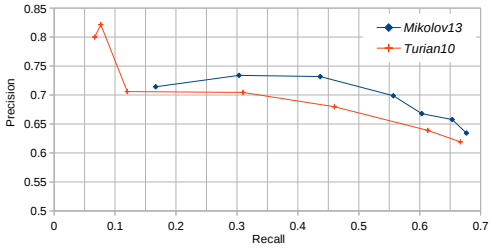


Figure 8: Precision-Recall curve.

has a corresponding path of length ≤ 5 in $\mathbf{T}$ with a similarity score > 0.4.

5 Conclusion and Discussion

We have presented a method of deriving abstract denotation from DCS trees, which enables logical inference on DCS, and we developed a textual inference system based on the framework. Experimental results have shown the power of the representation that allows both strict inference as on FraCaS data and robust reasoning as on RTE data.

Exploration of an appropriate meaning representation for querying and reasoning on knowledge bases has a long history. Description logic, being less expressive than FOL but featuring more efficient reasoning, is used as a theory base for Semantic Web (W3C, 2012). Ideas similar to our framework, including the use of sets in a representation that benefits efficient reasoning, are also found in description logic and knowledge representation community (Baader et al., 2003; Sowa, 2000; Sukkarieh, 2003). To our knowledge, however, their applications to logical inference beyond the use for database querying have not been much explored in the context of NLP.

The pursue of a logic more suitable for natural language inference is not new. For instance, MacCartney and Manning (2008) has implemented a model of natural logic (Lakoff, 1970). While being computationally efficient, various inference patterns are out of the scope of their system.

Much work has been done in mapping natural language into database queries (Cai and Yates, 2013; Kwiatkowski et al., 2013; Poon, 2013). Among these, the (λ-)DCS (Liang et al., 2011; Berant et al., 2013) framework defines algorithms that transparently map a labeled tree to a database querying procedure. Essentially, this is because DCS trees restrict the querying process to a very limited subset of possible operations. Our main contribution, the abstract denotation of DCS trees, can thus be considered as an attempt to characterize a fragment of FOL that is suited for both natural language inference *and* transparent syntax-semantics mapping, through the choice of operations and relations on sets.

We have demonstrated the utility of logical inference on DCS through the RTE task. A wide variety of strategies tackling the RTE task have been investigated (Androutsopoulos and Malakasiotis, 2010), including the comparison of surface strings (Jijkoun and De Rijke, 2005), syntactic and semantic structures (Haghighi et al., 2005; Snow et al., 2006; Zanzotto et al., 2009; Burchardt et al., 2009; Heilman and Smith, 2010; Wang and Manning, 2010), semantic vectors (Erk and Padó, 2009) and logical representations (Bos and Markert, 2005; Raina et al., 2005; Tatu and Moldovan, 2005). Acquisition of basic knowledge for RTE is also a huge stream of research (Lin and Pantel, 2001; Shinyama et al., 2002; Sudo et al., 2003; Szpektor et al., 2004; Fujita et al., 2012; Weisman et al., 2012; Yan et al., 2013). These previous works include various techniques for acquiring and incorporating different kinds of linguistic and world knowledge, and further fight against the knowledge bottleneck problem, e.g. by back-off to shallower representations.

Logic-based RTE systems employ various approaches to bridge knowledge gaps. Bos and Markert (2005) proposes features from a model builder; Raina et al. (2005) proposes an abduction process; Tatu and Moldovan (2006) shows handcrafted rules could drastically improve the performance of a logic-based RTE system.

As such, our current RTE system is at a proof-of-concept stage, in that many of the above techniques are yet to be implemented. Nonetheless, we would like to emphasize that it already shows performance competitive to state-of-the-art systems on one data set (RTE5). Other directions of our future work include further exploitation of the new semantic representation. For example, since abstract denotations are readily suited for data querying, they can be used to verify newly generated assumptions by fact search in a database. This may open a way towards a hybrid approach to RTE wherein logical inference is intermingled with large scale database querying.

Acknowledgments This research was supported by the Todai Robot Project at National Institute of Informatics.

References

Ion Androutsopoulos and Prodromos Malakasiotis. 2010. A survey of paraphrasing and textual entailment methods. *J. Artif. Int. Res.*, 38(1).

Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, USA.

Roy Bar-Haim, Ido Dagan, Iddo Greental, and Eyal Shnarch. 2007. Semantic inference at the lexical-syntactic level. In *Proceedings of AAAI 2007*.

Islam Beltagy, Cuong Chau, Gemma Boleda, Dan Garrette, Katrin Erk, and Raymond Mooney. 2013. Montague meets markov: Deep semantics with probabilistic logical form. In *Second Joint Conference on Lexical and Computational Semantics (*SEM)*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of EMNLP 2013*.

Johan Bos and Katja Markert. 2005. Recognising textual entailment with logical inference. In *Proceedings of EMNLP 2005*.

Johan Bos and Katja Markert. 2006. When logical inference helps determining textual entailment (and when it doesnt). In *Proceedings of the 2nd PASCAL RTE Challenge Workshop*.

Aljoscha Burchardt, Marco Pennacchiotti, Stefan Thater, and Manfred Pinkal. 2009. Assessing the impact of frame semantics on textual entailment. *Nat. Lang. Eng.*, 15(4).

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of ACL 2013*.

Peter Clark and Phil Harrison. 2008. Recognizing textual entailment with logical inference. In *Proceedings of 2008 Text Analysis Conference (TAC'08)*.

Peter Clark and Phil Harrison. 2009. Large-scale extraction and use of knowledge from text. In *Proceedings of the Fifth International Conference on Knowledge Capture (K-CAP'09)*.

E. F. Codd. 1970. A relational model of data for large shared data banks. *Commun. ACM*, 13(6).

Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Johan Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, and et al. 1996. Using the framework. *FraCaS Deliverable D*, 16.

Ido Dagan, O. Glickman, and B. Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*.

Georgiana Dinu and Rui Wang. 2009. Inference rules and their application to recognizing textual entailment. In *Proceedings of EACL 2009*.

Katrin Erk and Sebastian Padó. 2009. Paraphrase assessment in structured vector space: Exploring parameters and datasets. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*.

Atsushi Fujita, Pierre Isabelle, and Roland Kuhn. 2012. Enlarging paraphrase collections through generalization and instantiation. In *Proceedings of EMNLP 2012*.

Aria Haghighi, Andrew Ng, and Christopher Manning. 2005. Robust textual inference via graph matching. In *Proceedings of EMNLP 2005*.

Michael Heilman and Noah A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Proceedings of NAACL 2010*.

Valentin Jijkoun and Maarten De Rijke. 2005. Recognizing textual entailment: Is word similarity enough? In *Machine Learning Challenge Workshop, volume 3944 of LNCS, Springer.*

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of EMNLP 2013*.

George Lakoff. 1970. Linguistics and natural logic. *Synthese*, 22(1-2).

Mike Lewis and Mark Steedman. 2013. Combined distributional and logical semantics. *Transactions of ACL*, 1.

Percy Liang, Michael Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of ACL 2011*.

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question-answering. *Nat. Lang. Eng.*, 7(4).

Bill MacCartney and Christopher D. Manning. 2007. Natural logic for textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

Bill MacCartney and Christopher D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of Coling 2008*.

Oren Melamud, Jonathan Berant, Ido Dagan, Jacob Goldberger, and Idan Szpektor. 2013. A two level model for context sensitive inference rules. In *Proceedings of ACL 2013*.

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

Patrick Pantel, Rahul Bhagat, Bonaventura Coppola, Timothy Chklovski, and Eduard Hovy. 2007. ISP: Learning inferential selectional preferences. In *Proceedings of NAACL 2007*.

Hoifung Poon. 2013. Grounded unsupervised semantic parsing. In *Proceedings of ACL 2013*.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nate Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A multi-pass sieve for coreference resolution. In *Proceedings of EMNLP 2010*.

Rajat Raina, Andrew Y. Ng, and Christopher D. Manning. 2005. Robust textual inference via learning and abductive reasoning. In *Proceedings of AAAI 2005*.

Yusuke Shinyama, Satoshi Sekine, and Kiyoshi Sudo. 2002. Automatic paraphrase acquisition from news articles. In *Proceedings of HLT 2002*.

Rion Snow, Lucy Vanderwende, and Arul Menezes. 2006. Effectively using syntax for recognizing false entailment. In *Proceedings of NAACL 2006*.

Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. 2013. Parsing with compositional vector grammars. In *Proceedings of ACL 2013*.

John F. Sowa. 2000. *Knowledge Representation: Logical, Philosophical and Computational Foundations*. Brooks/Cole Publishing Co., Pacific Grove, CA, USA.

Asher Stern and Ido Dagan. 2011. A confidence model for syntactically-motivated entailment proofs. In *Proceedings of RANLP 2011*.

Asher Stern, Roni Stern, Ido Dagan, and Ariel Felner. 2012. Efficient search for transformation-based inference. In *Proceedings of ACL 2012*.

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. 2003. An improved extraction pattern representation model for automatic ie pattern acquisition. In *Proceedings of ACL 2003*.

JanaZ. Sukkarieh. 2003. An expressive efficient representation: Bridging a gap between nlp and kr. In Vasile Palade, RobertJ. Howlett, and Lakhmi Jain, editors, *Knowledge-Based Intelligent Information and Engineering Systems*. Springer Berlin Heidelberg.

Idan Szpektor, Hristo Tanev, Ido Dagan, and Bonaventura Coppola. 2004. Scaling web-based acquisition of entailment relations. In *Proceedings of EMNLP 2004*.

Idan Szpektor, Eyal Shnarch, and Ido Dagan. 2007. Instance-based evaluation of entailment rule acquisition. In *Proceedings of ACL 2007*.

Marta Tatu and Dan Moldovan. 2005. A semantic approach to recognizing textual entailment. In *Proceedings of EMNLP 2005*.

Marta Tatu and Dan Moldovan. 2006. A logic-based semantic approach to recognizing textual entailment. In *Proceedings of the COLING/ACL 2006*.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of ACL 2010*.

W3C. 2012. Owl 2 web ontology language document overview (second edition). www.w3.org/TR/owl2-overview/.

Mengqiu Wang and Christopher Manning. 2010. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proceedings of Coling 2010*.

Hila Weisman, Jonathan Berant, Idan Szpektor, and Ido Dagan. 2012. Learning verb inference rules from linguistically-motivated evidence. In *Proceedings of EMNLP 2012*.

Yulan Yan, Chikara Hashimoto, Kentaro Torisawa, Takao Kawai, Jun'ichi Kazama, and Stijn De Saeger. 2013. Minimally supervised method for multilingual paraphrase extraction from definition sentences on the web. In *Proceedings of NAACL 2013*.

Fabio massimo Zanzotto, Marco Pennacchiotti, and Alessandro Moschitti. 2009. A machine learning approach to textual entailment recognition. *Nat. Lang. Eng.*, 15(4).

A practical and linguistically-motivated approach to compositional distributional semantics

Denis Paperno and **Nghia The Pham** and **Marco Baroni**
Center for Mind/Brain Sciences (University of Trento, Italy)
(denis.paperno|thenghia.pham|marco.baroni)@unitn.it

Abstract

Distributional semantic methods to approximate word meaning with context vectors have been very successful empirically, and the last years have seen a surge of interest in their compositional extension to phrases and sentences. We present here a new model that, like those of Coecke et al. (2010) and Baroni and Zamparelli (2010), closely mimics the standard Montagovian semantic treatment of composition in distributional terms. However, our approach avoids a number of issues that have prevented the application of the earlier linguistically-motivated models to full-fledged, real-life sentences. We test the model on a variety of empirical tasks, showing that it consistently outperforms a set of competitive rivals.

1 Compositional distributional semantics

The research of the last two decades has established empirically that distributional vectors for words obtained from corpus statistics can be used to represent word meaning in a variety of tasks (Turney and Pantel, 2010). If distributional vectors encode certain aspects of word meaning, it is natural to expect that similar aspects of sentence meaning can also receive vector representations, obtained compositionally from word vectors. Developing a practical model of compositionality is still an open issue, which we address in this paper. One approach is to use simple, parameter-free models that perform operations such as pointwise multiplication or summing (Mitchell and Lapata, 2008). Such models turn out to be surprisingly effective in practice (Blacoe and Lapata, 2012), but they have obvious limitations. For instance, symmetric operations like vector addition are insensitive to syntactic structure, therefore meaning differences encoded in word order are lost in composition: *pandas eat bamboo* is identical to *bamboo eats pandas*. Guevara (2010), Mitchell and Lapata (2010), Socher et al. (2011) and Zanzotto et al. (2010) generalize the simple additive model by applying structure-encoding operators to the vectors of two sister nodes before addition, thus breaking the inherent symmetry of the simple additive model. A related approach (Socher et al., 2012) assumes richer lexical representations where each word is represented with a vector and a matrix that encodes its interaction with its syntactic sister. The training proposed in this model estimates the parameters in a supervised setting. Despite positive empirical evaluation, this approach is hardly practical for general-purpose semantic language processing, since it requires computationally expensive approximate parameter optimization techniques, and it assumes task-specific parameter learning whose results are not meant to generalize across tasks.

1.1 The lexical function model

None of the proposals mentioned above, from simple to elaborate, incorporates in its architecture the intuitive idea (standard in theoretical linguistics) that semantic composition is more than a weighted combination of words. Generally one of the components of a phrase, e.g., an adjective, acts as a function affecting the other component (e.g., a noun). This underlying intuition, adopted from formal semantics of natural language, motivated the creation of the *lexical function* model of composition (*lf*) (Baroni and Zamparelli, 2010; Coecke et al., 2010). The lf model can be seen as a projection of the symbolic Montagovian approach to semantic composition in natural language onto the domain of vector spaces and linear operations on them (Baroni et al., 2013). In lf, arguments are vectors and functions taking arguments (e.g., adjectives that combine with nouns) are tensors, with the number of arguments (n) determining the

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 90–99,
Baltimore, Maryland, USA, June 23-25 2014.

order of tensor (n+1). For example, adjectives, as unary functors, are modeled with 2-way tensors, or matrices. Tensor by vector multiplication formalizes function application and serves as the general composition method.

Baroni and Zamparelli (2010) propose a practical and empirically effective way to estimate matrices representing adjectival modifiers of nouns by linear regression from corpus-extracted examples of noun and adjective-noun vectors. Unlike the neural network approach of Socher et al. (2011; 2012), the Baroni and Zamparelli method does not require manually labeled data nor costly iterative estimation procedures, as it relies on automatically extracted phrase vectors and on the analytical solution of the least-squares-error problem.

The same method was later applied to matrix representations of intransitive verbs and determiners (Bernardi et al., 2013; Dinu et al., 2013), always with good empirical results.

The full range of semantic types required for natural language processing, including those of adverbs and transitive verbs, has to include, however, tensors of greater rank. The estimation method originally proposed by Baroni and Zamparelli has been extended to 3-way tensors representing transitive verbs by Grefenstette et al. (2013) with preliminary success. Grefenstette et al.'s method works in two steps. First, one estimates matrices of verb-object phrases from subject and subject-verb-object vectors; next, transitive verb tensors are estimated from verb-object matrices and object vectors.

1.2 Problems with the extension of the lexical function model to sentences

With all the advantages of lf, scaling it up to arbitrary sentences, however, leads to several issues. In particular, it is desirable for all practical purposes to limit representation size. For example, if noun meanings are encoded in vectors of 300 dimensions, adjectives become matrices of 300^2 cells, and transitive verbs are represented as tensors with 300^3=$27,000,000$ dimensions.

Estimating tensors of this size runs into data sparseness issues already for less common transitive verbs. Indeed, in order to train a transitive verb tensor (e.g., *eat*), the method of Grefenstette et al. (2013) requires a sufficient number of distinct verb object phrases with that verb (e.g., *eat cake*, *eat fruits*), each attested in combination with a certain number of subject nouns with sufficient frequency to extract sensible vectors. It is not feasible to obtain enough data points for all verbs in such a training design.

Things get even worse for other categories. Adverbs like *quickly* that modify intransitive verbs have to be represented with 300^{2^2} = $8,100,000,000$ dimensions. Modifiers of transitive verbs would have even greater representation size, which may not be possible to store and learn efficiently.

Another issue is that the same or similar items that occur in different syntactic contexts are assigned different semantic types with incomparable representations. For example, verbs like *eat* can be used in transitive or intransitive constructions (*children eat meat*/*children eat*), or in passive (*meat is eaten*). Since predicate arity is encoded in the order of the corresponding tensor, *eat* and the like have to be assigned different representations (matrix or tensor) depending on the context. Deverbal nouns like *demolition*, often used without mention of who demolished what, would have to get vector representations while the corresponding verbs (*demolish*) would become tensors, which makes immediately related verbs and nouns incomparable. Nouns in general would oscillate between vector and matrix representations depending on argument vs. predicate vs. modifier position (*an animal runs* vs. *this is an animal* vs. *animal shelter*). Prepositions are the hardest, as the syntactic positions in which they occur are most diverse (*park in the dark* vs. *play in the dark* vs. *be in the dark* vs. *a light glowing in the dark*).

In all those cases, the same word has to be mapped to tensors of different orders. Since each of these tensors must be learned from examples individually, their obvious relation is missed. Besides losing the comparability of the semantic contribution of a word across syntactic contexts, we also worsen the data sparseness issues.

The last, and related, point is that for the tensor calculus to work, one needs to model, for each word, each of the constructions in the corpus that the word is attested in. In its pure form lf does not include an emergency backoff strategy when unknown words or constructions are encountered. For example, if we only observe transitive usages of *to eat* in the training corpus, and encounter an intransitive or passive example of it in testing data,

the system would not be able to compose a sentence vector at all. This issue is unavoidable since we don't expect to find all words in all possible constructions even in the largest corpus.

2 The practical lexical function model

As follows from section 1.2, it would be desirable to have a compositional distributional model that encodes function-argument relations but avoids the troublesome high-order tensor representations of the pure lexical function model, with all the practical problems that come with them. We may still want to represent word meanings in different syntactic contexts differently, but at the same time we need to incorporate a formal connection between those representations, e.g., between the transitive and the intransitive instantiations of the verb *to eat*. Last but not least, all items need to include a common aspect of their representation (e.g., a vector) to allow comparison across categories (the case of *demolish* and *demolition*).

To this end, we propose a new model of composition that maintains the idea of function application, while avoiding the complications and rigidity of lf. We call our proposal *practical lexical function* model, or *plf*. In plf, a functional word is not represented by a single tensor of arity-dependent order, but by a vector plus an ordered set of matrices, with one matrix for each argument the function takes. After applying the matrices to the corresponding argument vectors, a single representation is obtained by summing across all resulting vectors.

2.1 Word meaning representation

In plf, all words are represented by a vector, and functional words, such as predicates and modifiers, are also assigned one or more matrices. The general form of a semantic representation for a linguistic unit is an ordered tuple of a vector and $n \in \mathbb{N}$ matrices:[1]

$$\left\langle \vec{x}, \overset{\square}{x}_1, \ldots, \overset{\square}{x}_n \right\rangle$$

The number of matrices in the representation encodes the arity of a linguistic unit, i.e., the number of other units to which it applies as a function. Each matrix corresponds to a function-argument relation, and words have as many matrices as many arguments they take: none for (most) nouns, one for adjectives and intransitive verbs, two for transitives, etc. The matrices formalize argument slot saturation, operating on an argument vector representation through matrix by vector multiplication, as described in the next section.

[1] Matrices associated with term x are symbolized $\overset{\square}{x}$.

dog	$\vec{dog}$
run	$\vec{run}, \overset{\square}{run}$
chase	$\vec{chase}, \overset{\square_s}{chase}, \overset{\square_o}{chase}$
give	$\vec{give}, \overset{\square_s}{give}, \overset{\square_o}{give}, \overset{\square_{io}}{give}$
big	$\vec{big}, \overset{\square}{big}$
very	$\vec{very}, \overset{\square_n}{very}, \overset{\square_a}{very}$
quickly	$\vec{quickly}, \overset{\square_s}{quickly}, \overset{\square_v}{quickly}$

Table 1: Examples of word representations. Subscripts encode, just for mnemonic purposes, the constituent whose vector the matrix combines with: **s**ubject, **o**bject, **i**ndirect **o**bject, **n**oun, **a**djective, **v**erb phrase.

Modifiers of n-ary functors are represented by n+1-ary structures. For instance, we treat adjectives that modify nouns (0-ary) as unary functions, encoded in a vector-matrix pair. Adverbs have different semantic types depending on their syntactic role. Sentential adverbs are unary, while adverbs that modify adjectives (*very*) or verb phrases (*quickly*) are encoded as binary functions, represented by a vector and two matrices. The form of semantic representations we are using is shown in Table 1.[2]

2.2 Semantic composition

Our system incorporates semantic composition via two composition rules, one for combining structures of different arity and the other for symmetric composition of structures with the same arity. These rules incorporate insights of two empirically successful models, lexical function and the simple additive approach, used as the default structure merging strategy.

The first rule is *function application*, illustrated in Figure 1. Table 2 illustrates simple cases of function application. For transitive verbs semantic composition applies iteratively as shown in the derivation of Figure 2. For ternary predicates such

[2] To determine the number and ordering of matrices representing the word in the current syntactic context, our plf implementation relies on the syntactic type assigned to the word in the categorial grammar parse of the sentence.

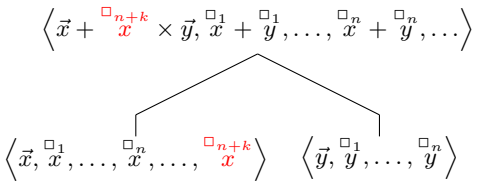


Figure 1: Function application: If two syntactic sisters have different arity, treat the higher-arity sister as the functor. Compose by multiplying the last matrix in the functor tuple by the argument vector and summing the result to the functor vector. Unsaturated matrices are carried up to the composed node, summing across sisters if needed.

dogs	$\vec{dogs}$
run	$\vec{run}, \square run$
dogs run	$\vec{run} + \square run \times \vec{dog}$
house	$\vec{house}$
big	$\vec{big}, \square big$
big house	$\vec{big} + \square big \times \vec{house}$

Table 2: Examples of function application.

as *give* in a ditransitive construction, the first step in the derivation absorbs the innermost argument by multiplying its vector by the third *give* matrix, and then composition proceeds like for transitives.

The second composition rule, *symmetric composition* applies when two syntactic sisters are of the same arity (e.g., two vectors, or two vector-matrix pairs). Symmetric composition simply sums the objects in the two tuples: vector with vector, n-th matrix with n-th matrix.

Symmetric composition is reserved for structures in which the function-argument distinction is problematic. Some candidates for such treatment are coordination and nominal compounds, although we recognize that the headless analysis is

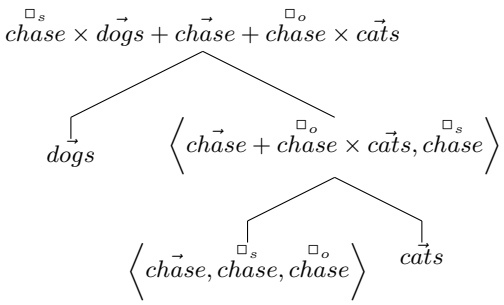


Figure 2: Applying function application twice to derive the representation of a transitive sentence.

sing: $\vec{sing}, \square sing$	dance: $\vec{dance}, \square dance$
sing and dance:	$\vec{sing} + \vec{dance}, \square sing + \square dance$
rice: $\vec{rice}$	cake: $\vec{cake}$
rice cake	$\vec{rice} + \vec{cake}$

Table 3: Examples of symmetric composition.

not the only possible one here. See two examples of Symmetric Composition application in Table 3.

Note that the *sing and dance* composition in Table 3 skips the conjunction. Our current plf implementation treats most grammatical words, including conjunctions, as "empty" elements, that do not project into semantics. This choice leads to some interesting "serendipitous" treatments of various constructions. For example, since the copula is empty, a sentence with a predicative adjective (*cars are red*) is treated in the same way as a phrase with the same adjective in attributive position (*red cars*) – although the latter, being a phrase and not a full sentence, will later be embedded as argument in a larger construction. Similarly, leaving the relative pronoun empty makes *cars that run* identical to *cars run*, although, again, the former will be embedded in a larger construction later in the derivation.

We conclude our brief exposition of plf with an alternative intuition for it: the plf model is also a more sophisticated version of the additive approach, where argument words are adapted by matrices that encode the relation to their functors before the sentence vector is derived by summing.

2.3 Satisfying the desiderata

Let us now outline how plf addresses the shortcomings of lf listed in Section 1.2. First, all issues caused by representation size disappear. An n-ary predicate is no longer encoded as an n+1-way tensor; instead we have a sequence of n matrices. The representation size grows linearly, not exponentially, for higher semantic types, allowing for simpler and more efficient parameter estimation, storage, and computation.

As a consequence of our architecture, we no longer need to perform the complicated step-by-step estimation for elements of higher arity. Indeed, one can estimate each matrix of a complex representation individually using the simple method of Baroni and Zamparelli (2010). For instance, for transitive verbs we estimate the verb-subject combination matrix from subject and verb-

boys	$\vec{boys}$
eat (intrans.)	$\vec{eat}, \overset{\square_s}{eat}$
boys eat	$\overset{\square_s}{eat} \times \vec{boys} + \vec{eat}$
meat	$\vec{meat}$
eat (trans.)	$\vec{eat}, \overset{\square_s}{eat}, \overset{\square_o}{eat}$
boys eat meat	$\overset{\square_s}{eat} \times \vec{boys} + \vec{eat} + \overset{\square_o}{eat} \times \vec{meat}$
(is) eaten (pass.)	$\vec{eat}, \overset{\square_o}{eat}$
meat is eaten	$\vec{eat} + \overset{\square_o}{eat} \times \vec{meat}$

Table 4: The verb *to eat* associated to different sets of matrices in different syntactic contexts.

subject vectors, the verb-object combination matrix from object and verb-object vectors. We expect a reasonably large corpus to feature many occurrences of a verb with a variety of subjects and a variety of objects (but not necessarily a variety of subjects with each of the objects as required by Grefenstette et al.'s training), allowing us to avoid the data sparseness issue.

The semantic representations we propose include a semantic vector for constituents of any semantic type, thus enabling semantic comparison for words of different parts of speech (the case of *demolition* vs. *demolish*).

Finally, the fact that we represent the predicate interaction with each of its arguments in a separate matrix allows for a natural and intuitive treatment of argument alternations. For instance, as shown in Table 4, one can distinguish the transitive and intransitive usages of the verb *to eat* by the presence of the object-oriented matrix of the verb while keeping the rest of the representation intact. To model passive usages, we insert the object matrix of the verb only, which will be multiplied by the syntactic subject vector, capturing the similarity between *eat meat* and *meat is eaten*.

So keeping the verb's interaction with subject and object encoded in distinct matrices not only solves the issues of representation size for arbitrary semantic types, but also provides a sensible built-in strategy for handling a word's occurrence in multiple constructions. Indeed, if we encounter a verb used intransitively which was only attested as transitive in the training corpus, we can simply omit the object matrix to obtain a type-appropriate representation. On the other hand, if the verb occurs with more arguments than usual in testing materials, we can add a default diagonal identity matrix to its representation, signaling agnosticism about how the verb relates to the unexpected argument. This flexibility makes our model suitable to compute vector representations of sentences without stumbling at unseen syntactic usages of words.

To summarize, plf is an extension of the lexical function model that inherits its strengths and overcomes its weaknesses. We still employ a linguistically-motivated notion of semantic composition as function application and use distinct kinds of representations for different semantic types. At the same time, we avoid high order tensor representations, produce semantic vectors for all syntactic constituents, and allow for an elegant and transparent correspondence between different syntactic usages of a lexeme, such as the transitive, the intransitive, and the passive usages of the verb *to eat*. Last but not least, our implementation is suitable for realistic language processing since it allows to produce vectors for sentences of arbitrary size, including those containing novel syntactic configurations.

3 Evaluation

3.1 Evaluation materials

We consider 5 different benchmarks that focus on different aspects of sentence-level semantic composition. The first data set, created by Edward Grefenstette and Mehrnoosh Sadrzadeh and introduced in Kartsaklis et al. (2013), features 200 sentence pairs that were rated for similarity by 43 annotators. In this data set, sentences have fixed adjective-noun-verb-adjective-noun (anvan) structure, and they were built in order to crucially require context-based verb disambiguation (e.g., *young woman filed long nails* is paired with both *young woman smoothed long nails* and *young woman registered long nails*). We also consider a similar data set introduced by Grefenstette (2013), comprising 200 sentence pairs rated by 50 annotators. We will call these benchmarks **anvan1** and **anvan2**, respectively. Evaluation is carried out by computing the Spearman correlation between the annotator similarity ratings for the sentence pairs and the cosines of the vectors produced by the various systems for the same sentence pairs.

The benchmark introduced by The Pham et al. (2013) at the TFDS workshop (**tfds** below) was specifically designed to test compositional methods for their sensitivity to word order and the semantic effect of determiners. The tfds benchmark contains 157 target sentences that are matched with a set of (approximate) paraphrases (8 on av-

erage), and a set of "foils" (17 on average). The foils have high lexical overlap with the targets but very different meanings, due to different determiners and/or word order. For example, the target *A man plays an acoustic guitar* is matched with paraphrases such as *A man plays guitar* and *The man plays the guitar*, and foils such as *The man plays no guitar* and *A guitar plays a man*. A good system should return higher similarities for the comparison with the paraphrases with respect to that with the foils. Performance is assessed through the t-standardized cross-target average of the difference between mean cosine with paraphrases and mean cosine with foils (Pham and colleagues, equivalently, reported non-standardized average and standard deviations).

The two remaining data sets are larger and more 'natural', as they were not constructed by linguists under controlled conditions to focus on specific phenomena. They are aimed at evaluating systems on the sort of free-form sentences one encounters in real-life applications. The **msrvid** data set from the SemEval-2012 Semantic Textual Similarity (STS) task (Agirre et al., 2012) consists of 750 sentence pairs that describe brief videos. Sentence pairs were scored for similarity by 5 subjects each. Following standard practice in paraphrase detection studies (e.g., Blacoe and Lapata (2012)), we use cosine similarity between sentence pairs as computed by one of our systems together with two shallow similarity cues: word overlap between the two sentences and difference in sentence length. We obtain a final similarity score by weighted addition of the 3 cues, with the optimal weights determined by linear regression on separate msrvid train data that were also provided by the SemEval task organizers (before combining, we checked that the collinearity between cues was low). System scores are evaluated by their Pearson correlation with the human ratings.

The final set we use is **onwn**, from the *SEM-2013 STS shared task (Agirre et al., 2013). This set contains 561 pairs of glosses (from the WordNet and OntoNotes databases), rated by 5 judges for similarity. Our main interest in this set stems from the fact that glosses are rarely well-formed full sentences (consider, e.g., *cause something to pass or lead somewhere*; *coerce by violence, fill with terror*). For this reason, they are very challenging for standard parsers. Indeed, we estimated from a sample of 40 onwn glosses that the C&C parser (see below) has only 45% accuracy on this set. Since *plf* needs syntactic information to construct sentence vectors compositionally, we test it on onwn to make sure that it is not overly sensitive to parser noise. Evaluation proceeds as with msrvid (cue weights are determined by 10-fold cross-validation).[3]

3.2 Semantic space construction and composition model implementation

Our source corpus was given by the concatenation of ukWaC (`wacky.sslmit.unibo.it`), a mid-2009 dump of the English Wikipedia (`en.wikipedia.org`) and the British National Corpus (`www.natcorp.ox.ac.uk`), for a total of about 2.8 billion words.

We collected a 30K-by-30K matrix by counting co-occurrence of the 30K most frequent content lemmas (nouns, adjectives and verbs) within a 3-word window. The raw count vectors were transformed into positive Pointwise Mutual Information scores and reduced to 300 dimensions by the Singular Value Decomposition. All vectors were normalized to length 1. This setup was picked without tuning, as we found it effective in previous, unrelated experiments.[4]

We consider four composition models. The **add** (additive) model produces the vector of a sentence by summing the vectors of all content words in it. Similarly, **mult** uses component-wise multiplication of vectors for composition. While these models are very simple, a long experimental tradition has proven their effectiveness (Landauer and Dumais, 1997; Mitchell and Lapata, 2008; Mitchell and Lapata, 2010; Blacoe and Lapata, 2012).

For the **lf** (lexical function) model, we construct functional matrix representations of adjectives, determiners and intransitive verbs. These are trained using Ridge regression with generalized cross-validation from corpus-extracted vectors of nouns,

[3] We did not evaluate on other STS benchmarks since they have characteristics, such as high density of named entities, that would require embedding our compositional models into more complex systems, obfuscating their impact on the overall performance.

[4] With the multiplicative composition model we also tried Nonnegative Matrix Factorization instead of Singular Value Decomposition, because the negative values produced by SVD are potentially problematic for mult. In addition, we repeated the evaluation for the multiplicative and additive models without any form of dimensionality reduction. The overall pattern of results did not change significantly, and thus for consistency we report all models' performance only for the SVD-reduced space.

as input, and phrases including those nouns as output (e.g., the matrix for *red* is trained from corpus-extracted $\langle noun, red\text{-}noun \rangle$ vector pairs). Transitive verb tensors are estimated using the two-step regression procedure outlined by Grefenstette et al. (2013). We did not attempt to train a lf model for the larger and more varied msrvid and onwn data sets, as this would have been extremely time consuming and impractical for all the reasons we discussed in Section 1.2 above.

Training **plf** (practical lexical function) proceeds similarly, but we also build preposition matrices (from $\langle noun, preposition\text{-}noun \rangle$ vector pairs), and for verbs we prepare separate subject and object matrices.

Since syntax guides lf and plf composition, we supplied all test sentences with categorial grammar parses. Every sentence in the anvan1 and anvan2 datasets has the form (subject) Adjective + Noun + Transitive Verb + (object) Adjective + Noun, so parsing them is trivial. All sentences in tfds have a predictable structure that allows perfect parsing with simple finite state rules. In all these cases, applying a general-purpose parser to the data would have, at best, had no impact and, at worst, introduced parsing errors. For msrvid and onwn, we used the output of the C&C parser (Clark and Curran, 2007).

3.3 Results

Table 5 summarizes the performance of our models on the chosen tasks, and compares it to the state of the art reported in previous work, as well as to various strong baselines.

The plf model performs very well on both anvan benchmarks, outperforming not only add and mult, but also the full-fledged lf model. Given that these data sets contain, systematically, transitive verbs, the major difference between plf and lf lies in their representation of the latter. Evidently, the separately-trained subject and object matrices of plf, being less affected by data sparseness than the 3-way tensors of lf, are better able to capture how verbs interact with their arguments. For anvan1, plf is just below the state of the art, which is based on disambiguating the verb vector in context (Kartsaklis and Sadrzadeh, 2013), and lf outperforms the baseline, which consists in using the verb vector only as a proxy to sentence similarity.[5] On anvan2, plf outperforms the best model reported by Grefenstette (2013) (an implementation of the lexical function ideas along the lines of Grefenstette and Sadrzadeh (2011a; 2011b)). And lf is, again, the only model, besides plf, that performs better than the baseline.

models	anvan 1	anvan 2	tfds	msr vid	onwn
add	8	22	-0.2	78	66
mult	8	-4	-2.3	77	55
lf	15	30	**5.90**	NA	NA
plf	**20**	**36**	2.7	**79**	**67**
soa	22	27	11.4	87	75
baseline	8	22	7.9	77	55

Table 5: Performance of composition models on all evaluation sets. Figures of merit follow previous art on each set and are: percentage Spearman coefficients for anvan1 and anvan2, t-standardized average difference between mean cosines with paraphrases and with foils for tfds, percentage Pearson coefficients for msrvid and onwn. State-of-the-art (soa) references: anvan1: Kartsaklis and Sadrzadeh (2013); anvan2: Grefenstette (2013); tfds: The Pham et al. (2013); msrvid: Bär et al. (2012); onwn: Han et al. (2013). Baselines: anvan1/anvan2: verb vectors only; tfds: word overlap; msrvid/onwn: word overlap + sentence length.

In the tfds task, not surprisingly the add and mult models, lacking determiner representations and being order-insensitive, fail to distinguish between true paraphrases and foils (indeed, for the mult model foils are significantly *closer* to the targets than the paraphrases, probably because the latter have lower content word overlap than the foils, that often differ in word order and determiners only). Our plf approach is able to handle determiners and word order correctly, as demonstrated by a highly significant ($p < 0.01$) difference between paraphrase and foil similarity (average difference in cosine .017, standard deviation .077). In this case, however, the traditional lf model (average difference .044, standard deviation .092) outperforms plf. Since determiners are handled identically under the two approaches, the culprit must be word order. We conjecture that the lf 3-way tensor representation of transitive verbs leads to a stronger asymmetry between sentences with in-

[5] We report state of the art from Kartsaklis and Sadrzadeh (2013) rather than Kartsaklis et al. (2013), since only the former used a source corpus that is comparable to ours.

verted arguments, and thus makes this model particularly sensitive to word order differences. Indeed, if we limit evaluation to those foils characterized by word order changes only, lf discriminates between paraphrases and foils even more clearly, whereas the plf difference, while still significant, decreases slightly.

The state-of-the-art row for tfds reports the lf implementation by The Pham et al. (2013), which outperforms ours. The main difference is that Pham and colleagues do not normalize vectors like we do. If we don't normalize, we do get larger differences for our models as well, but consistently lower performance in all other tasks. More worryingly, the simple word overlap baseline reported in the table sports a larger difference than our best model. Clearly, this baseline is exploiting the systematic determiner differences in the foils and, indeed, when it is evaluated on foils where only word order changes its performance is no longer significant.

On msrvid, the plf approach outperforms add and mult, although the difference between the three is not big. Our result stands in contrast with Blacoe and Lapata (2012), the only study we are aware of that compared a sophisticated composition model (Socher et al.'s 2011 model) to add and mult on realistic sentences, which attained the top performance with the simple models for both figures of merit they used.[6] The best 2012 STS system (Bär et al., 2012), obtained 0.87 correlation, but with many more and considerably more complex features than the ones we used here. Indeed, our simple system would have obtained a respectable 25/89 ranking in the STS 2012 msrvid task. Still, we must also stress the impressive performance of our baseline, given by the combination of the word overlap and sentence length cues. This suggests that the msrvid benchmark lacks the lexical and syntactic variety we would like to test our systems on.

Our plf model is again the best on the onwn set (albeit by a small margin over add). This is a very positive result, in the light of the fact that the parser has very low performance on the onwn glosses, thus suggesting that plf can produce sensible semantic vectors from noisy syntactic representations. Here the overlap+length baseline does not perform so well, and again the best STS 2013 system (Han et al., 2013) uses considerably richer knowledge sources and algorithms than ours. Our plf-based method would have reached a respectable 20/90 rank in the STS 2013 onwn task.

As a final remark, in all experiments the running time of plf was only slightly larger than for the simpler models, but orders of magnitude smaller than lf, confirming another practical side of our approach.

4 Conclusion

We introduced an approach to compositional distributional semantics based on a linguistically-motivated syntax-to-semantics type mapping, but simple and flexible enough that it can produce representations of English sentences of arbitrary size and structure.

We showed that our approach is competitive against the more complex lexical function model when evaluated on the simple constructions the latter can be applied to, and it outperforms the additive and multiplicative compositionality models when tested on more realistic benchmarks (where the full-fledged lexical function approach is difficult or impossible to use), even in presence of strong noise in its syntactic input. While our results are encouraging, no current benchmark combines large-scale, real-life data with the syntactic variety on which a syntax-driven approach to semantics such as ours could truly prove its worth. The recently announced SemEval 2014 Task 1[7] is filling exactly this gap, and we look forward to apply our method to this new benchmark, as soon as it becomes available.

One of the strengths of our framework is that it allows for incremental improvement focused on specific constructions. For example, one could add representations for different conjunctions (*and* vs. *or*), train matrices for verb arguments other than subject and direct object, or include new types of modifiers into the model, etc.

While there is potential for local improvements, our framework, which extends and improves on existing compositional semantic vector models, has demonstrated its ability to account for full sentences in a principled and elegant way. Our implementation of the model relies on simple and effi-

[6] We refer here to the results reported in the erratum available at `http://homepages.inf.ed.ac.uk/s1066731/pdf/emnlp2012erratum.pdf`. The add/mult advantage was even more marked in the original paper.

[7] `http://alt.qcri.org/semeval2014/task1/`

cient training, works fast, and shows good empirical results.

Acknowledgements

We thank Roberto Zamparelli and the COMPOSES team for helpful discussions. This research was supported by the ERC 2011 Starting Independent Research Grant n. 283554 (COMPOSES).

References

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 Task 6: a pilot on semantic textual similarity. In *Proceedings of *SEM*, pages 385–393, Montreal, Canada.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic Textual Similarity. In *Proceedings of *SEM*, pages 32–43, Atlanta, GA.

Daniel Bär, Chris Biemann, Iryna Gurevych, and Torsten Zesch. 2012. UKP: Computing semantic textual similarity by combining multiple content similarity measures. In *Proceedings of *SEM*, pages 435–440, Montreal, Canada.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of EMNLP*, pages 1183–1193, Boston, MA.

Marco Baroni, Raffaella Bernardi, and Roberto Zamparelli. 2013. Frege in space: A program for compositional distributional semantics. *Linguistic Issues in Language Technology*. In press; `http://clic.cimec.unitn.it/composes/materials/frege-in-space.pdf`.

Raffaella Bernardi, Georgiana Dinu, Marco Marelli, and Marco Baroni. 2013. A relatedness benchmark to test the role of determiners in compositional distributional semantics. In *Proceedings of ACL (Short Papers)*, pages 53–57, Sofia, Bulgaria.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of EMNLP*, pages 546–556, Jeju Island, Korea.

Stephen Clark and James Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. *Linguistic Analysis*, 36:345–384.

Georgiana Dinu, Nghia The Pham, and Marco Baroni. 2013. General estimation and evaluation of compositional distributional semantic models. In *Proceedings of ACL Workshop on Continuous Vector Space Models and their Compositionality*, pages 50–58, Sofia, Bulgaria.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011a. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of EMNLP*, pages 1394–1404, Edinburgh, UK.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011b. Experimenting with transitive verbs in a DisCoCat. In *Proceedings of GEMS*, pages 62–66, Edinburgh, UK.

Edward Grefenstette, Georgiana Dinu, Yao-Zhong Zhang, Mehrnoosh Sadrzadeh, and Marco Baroni. 2013. Multi-step regression learning for compositional distributional semantics. In *Proceedings of IWCS*, pages 131–142, Potsdam, Germany.

Edward Grefenstette. 2013. *Category-Theoretic Quantitative Compositional Distributional Models of Natural Language Semantics*. PhD thesis, University of Oxford Essex.

Emiliano Guevara. 2010. A regression model of adjective-noun compositionality in distributional semantics. In *Proceedings of GEMS*, pages 33–37, Uppsala, Sweden.

Lushan Han, Abhay Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. 2013. UMBC_EBIQUITY-CORE: Semantic textual similarity systems. In *Proceedings of *SEM*, pages 44–52, Atlanta, GA.

Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. 2013. Prior disambiguation of word tensors for constructing sentence vectors. In *Proceedings of EMNLP*, pages 1590–1601, Seattle, WA.

Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. 2013. Separating disambiguation from composition in distributional semantics. In *Proceedings of CoNLL*, pages 114–123, Sofia, Bulgaria.

Thomas Landauer and Susan Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL*, pages 236–244, Columbus, OH.

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

Richard Socher, Eric Huang, Jeffrey Pennin, Andrew Ng, and Christopher Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of NIPS*, pages 801–809, Granada, Spain.

Richard Socher, Brody Huval, Christopher Manning, and Andrew Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP*, pages 1201–1211, Jeju Island, Korea.

Nghia The Pham, Raffaella Bernardi, Yao-Zhong Zhang, and Marco Baroni. 2013. Sentence paraphrase detection: When determiners and word order make the difference. In *Proceedings of the Towards a Formal Distributional Semantics Workshop at IWCS 2013*, pages 21–29, Potsdam, Germany.

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

Fabio Zanzotto, Ioannis Korkontzelos, Francesca Falucchi, and Suresh Manandhar. 2010. Estimating linear models for compositional distributional semantics. In *Proceedings of COLING*, pages 1263–1271, Beijing, China.

Lattice Desegmentation for Statistical Machine Translation

Mohammad Salameh[†] **Colin Cherry**[‡] **Grzegorz Kondrak**[†]

[†]Department of Computing Science
University of Alberta
Edmonton, AB, T6G 2E8, Canada
{msalameh,gkondrak}@ualberta.ca

[‡]National Research Council Canada
1200 Montreal Road
Ottawa, ON, K1A 0R6, Canada
Colin.Cherry@nrc-cnrc.gc.ca

Abstract

Morphological segmentation is an effective sparsity reduction strategy for statistical machine translation (SMT) involving morphologically complex languages. When translating into a segmented language, an extra step is required to desegment the output; previous studies have desegmented the 1-best output from the decoder. In this paper, we expand our translation options by desegmenting n-best lists or lattices. Our novel lattice desegmentation algorithm effectively combines both segmented and desegmented views of the target language for a large subspace of possible translation outputs, which allows for inclusion of features related to the desegmentation process, as well as an unsegmented language model (LM). We investigate this technique in the context of English-to-Arabic and English-to-Finnish translation, showing significant improvements in translation quality over desegmentation of 1-best decoder outputs.

1 Introduction

Morphological segmentation is considered to be indispensable when translating between English and morphologically complex languages such as Arabic. Morphological complexity leads to much higher type to token ratios than English, which can create sparsity problems during translation model estimation. Morphological segmentation addresses this issue by splitting surface forms into meaningful morphemes, while also performing orthographic transformations to further reduce sparsity. For example, the Arabic noun للدول *lldwl* "to the countries" is segmented as *l+* "to" *Aldwl* "the countries". When translating from Arabic, this segmentation process is performed as input preprocessing and is otherwise transparent to the translation system. However, when translating into Arabic, the decoder produces segmented output, which must be **desegmented** to produce readable text. For example, *l+ Aldwl* must be converted to *lldwl*.

Desegmentation is typically performed as a post-processing step that is independent from the decoding process. While this division of labor is useful, the pipeline approach may prevent the desegmenter from recovering from errors made by the decoder. Despite the efforts of the decoder's various component models, the system may produce mismatching segments, such as *s+ hzymp*, which pairs the future particle *s+* "will" with a noun *hzymp* "defeat", instead of a verb. In this scenario, there is no right desegmentation; the post-processor has been dealt a losing hand.

In this work, we show that it is possible to maintain the sparsity-reducing benefit of segmentation while translating directly into unsegmented text. We desegment a large set of possible decoder outputs by processing n-best lists or lattices, which allows us to consider both the segmented and desegmented output before locking in the decoder's decision. We demonstrate that significant improvements in translation quality can be achieved by training a linear model to re-rank this transformed translation space.

2 Related Work

Translating into morphologically complex languages is a challenging and interesting task that has received much recent attention. Most techniques approach the problem by transforming the target language in some manner before training the translation model. They differ in what transformations are performed and at what stage they are reversed. The transformation might take the form of a morphological analysis or a morphological segmentation.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 100–110,
Baltimore, Maryland, USA, June 23-25 2014.

2.1 Morphological Analysis

Many languages have access to morphological analyzers, which annotate surface forms with their lemmas and morphological features. Bojar (2007) incorporates such analyses into a factored model, to either include a language model over target morphological tags, or model the generation of morphological features. Other approaches train an SMT system to predict lemmas instead of surface forms, and then inflect the SMT output as a post-processing step (Minkov et al., 2007; Clifton and Sarkar, 2011; Fraser et al., 2012; El Kholy and Habash, 2012b). Alternatively, one can reparameterize existing phrase tables as exponential models, so that translation probabilities account for source context and morphological features (Jeong et al., 2010; Subotin, 2011). Of these approaches, ours is most similar to the translate-then-inflect approach, except we translate and then desegment. In particular, Toutanova et al. (2008) inflect and re-rank n-best lists in a similar manner to how we desegment and re-rank n-best lists or lattices.

2.2 Morphological Segmentation

Instead of producing an abstract feature layer, morphological segmentation transforms the target sentence by segmenting relevant morphemes, which are then handled as regular tokens during alignment and translation. This is done to reduce sparsity and to improve correspondence with the source language (usually English). Such a segmentation can be produced as a byproduct of analysis (Oflazer and Durgar El-Kahlout, 2007; Badr et al., 2008; El Kholy and Habash, 2012a), or may be produced using an unsupervised morphological segmenter such as Morfessor (Luong et al., 2010; Clifton and Sarkar, 2011). Work on target language morphological segmentation for SMT can be divided into three subproblems: segmentation, desegmentation and integration. Our work is concerned primarily with the integration problem, but we will discuss each subproblem in turn.

The usefulness of a target segmentation depends on its correspondence to the source language. If a morphological feature does not manifest itself as a separate token in the source, then it may be best to leave its corresponding segment attached to the stem. A number of studies have looked into what granularity of segmentation is best suited for a particular language pair (Oflazer and Durgar El-Kahlout, 2007; Badr et al., 2008; Clifton and Sarkar, 2011; El Kholy and Habash, 2012a). Since our focus here is on integrating segmentation into the decoding process, we simply adopt the segmentation strategies recommended by previous work: the Penn Arabic Treebank scheme for English-Arabic (El Kholy and Habash, 2012a), and an unsupervised scheme for English-Finnish (Clifton and Sarkar, 2011).

Desegmentation is the process of converting segmented words into their original surface form. For many segmentations, especially unsupervised ones, this amounts to simple concatenation. However, more complex segmentations, such as the Arabic tokenization provided by MADA (Habash et al., 2009), require further orthographic adjustments to reverse normalizations performed during segmentation. Badr et al. (2008) present two Arabic desegmentation schemes: table-based and rule-based. El Kholy and Habash (2012a) provide an extensive study on the influence of segmentation and desegmentation on English-to-Arabic SMT. They introduce an additional desegmentation technique that augments the table-based approach with an unsegmented language model. Salameh et al. (2013) replace rule-based desegmentation with a discriminatively-trained character transducer. In this work, we adopt the Table+Rules approach of El Kholy and Habash (2012a) for English-Arabic, while concatenation is sufficient for English-Finnish.

Work on integration attempts to improve SMT performance for morphologically complex target languages by going beyond simple pre- and post-processing. Oflazer and Durgar El-Kahlout (2007) desegment 1000-best lists for English-to-Turkish translation to enable scoring with an unsegmented language model. Unlike our work, they *replace* the segmented language model with the unsegmented one, allowing them to tune the linear model parameters by hand. We use both segmented and unsegmented language models, and tune automatically to optimize BLEU.

Like us, Luong et al. (2010) tune on unsegmented references,[1] and translate with both segmented and unsegmented language models for English-to-Finnish translation. However, they adopt a scheme of word-boundary-aware

[1]Tuning on unsegmented references does not require substantial modifications to the standard SMT pipeline. For example, Badr et al. (2008) also tune on unsegmented references by simply desegmenting SMT output before MERT collects sufficient statistics for BLEU.

morpheme-level phrase extraction, meaning that target phrases include only complete words, though those words are segmented into morphemes. This enables full decoder integration, where we do n-best and lattice re-ranking. But it also comes at a substantial cost: when target phrases include only complete words, the system can only generate word forms that were seen during training. In this setting, the sparsity reduction from segmentation helps word alignment and target language modeling, but it does not result in a more expressive translation model. Furthermore, it becomes substantially more difficult to have non-adjacent source tokens contribute morphemes to a single target word. For example, when translating "with his blue car" into the Arabic بسيارته الزرقاء *bsyArth AlzrqA'*, the target word *bsyArth* is composed of three tokens: *b+* "with", *syArp* "car" and *+h* "his". With word-boundary-aware phrase extraction, a phrase pair containing all of "with his blue car" must have been seen in the parallel data to translate the phrase correctly at test time. With lattice desegmentation, we need only to have seen *AlzrqA'* "blue" and the three morphological pieces of *bsyArth* for the decoder and desegmenter to assemble the phrase.

3 Methods

Our goal in this work is to benefit from the sparsity-reducing properties of morphological segmentation while simultaneously allowing the system to reason about the final surface forms of the target language. We approach this problem by augmenting an SMT system built over target segments with features that reflect the desegmented target words. In this section, we describe our various strategies for desegmenting the SMT system's output space, along with the features that we add to take advantage of this desegmented view.

3.1 Baselines

The two obvious baseline approaches each decode using one view of the target language. The **unsegmented** approach translates without segmenting the target. This trivially allows for an unsegmented language model and never makes desegmentation errors. However, it suffers from data sparsity and poor token-to-token correspondence with the source language.

The **one-best desegmentation** approach segments the target language at training time and then desegments the one-best output in post-processing. This resolves the sparsity issue, but does not allow the decoder to take into account features of the desegmented target. To the best of our knowledge, we are the first group to go beyond one-best desegmentation for English-to-Arabic translation. In English-to-Finnish, although alternative integration strategies have seen some success (Luong et al., 2010), the current state-of-the-art performs one-best-desegmentation (Clifton and Sarkar, 2011).

3.2 n-best Desegmentation

The one-best approach can be extended easily by desegmenting n-best lists of segmented decoder output. Doing so enables the inclusion of an unsegmented target language model, and with a small amount of bookkeeping, it also allows the inclusion of features related to the operations performed during desegmentation (see Section 3.4). With new features reflecting the desegmented output, we can re-tune our enhanced linear model on a development set. Following previous work, we will desegment 1000-best lists (Oflazer and Durgar El-Kahlout, 2007).

Once n-best lists have been desegmented, we can tune on unsegmented references as a side-benefit. This could improve translation quality, as it brings our training scenario closer to our test scenario (test BLEU is always measured on unsegmented references). In particular, it could address issues with translation length mismatch. Previous work that has tuned on unsegmented references has reported mixed results (Badr et al., 2008; Luong et al., 2010).

3.3 Lattice Desegmentation

An n-best list reflects a tiny portion of a decoder's search space, typically fixed at 1000 hypotheses. Lattices[2] can represent an exponential number of hypotheses in a compact structure. In this section, we discuss how a lattice from a multi-stack phrase-based decoder such as Moses (Koehn et al., 2007) can be desegmented to enable word-level features.

Finite State Analogy

A phrase-based decoder produces its output from left to right, with each operation appending the translation of a source phrase to a growing target hypothesis. Translation continues un-

[2]Or forests for hierarchical and syntactic decoders.

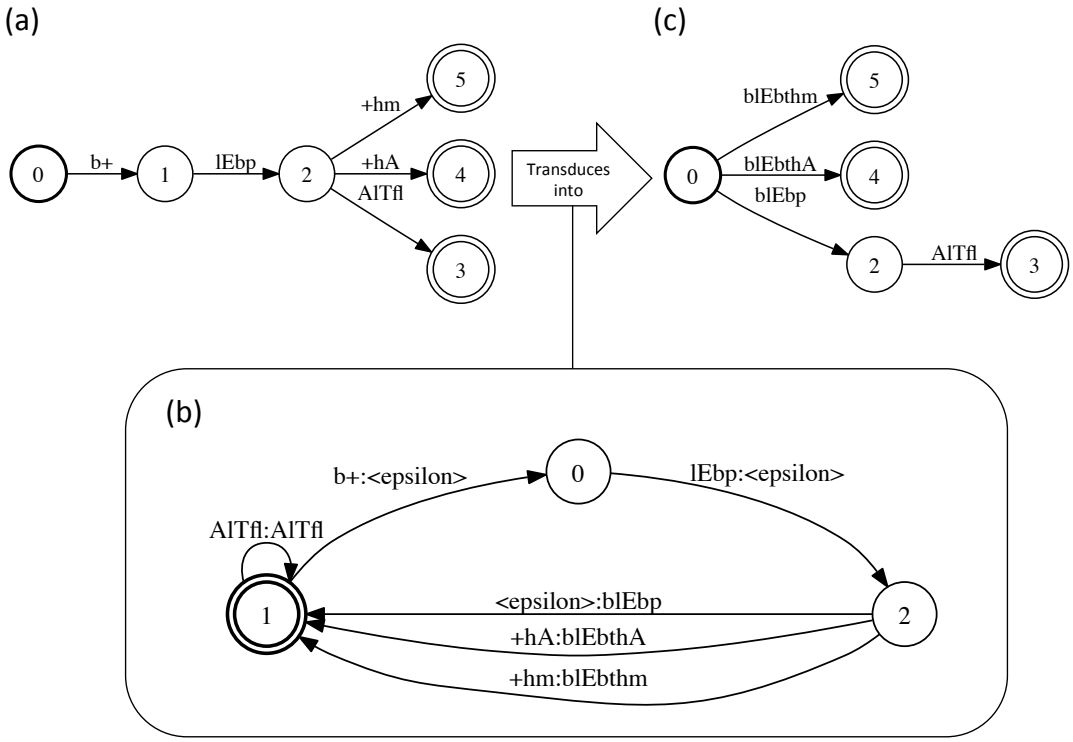


Figure 1: The finite state pipeline for a lattice translating the English fragment "with the child's game". The input morpheme lattice (a) is desegmented by composing it with the desegmenting transducer (b) to produce the word lattice (c). The tokens in (a) are: *b+* "with", *lEbp* "game", *+hm* "their", *+hA* "her", and *AlTfl* "the child".

til each source word has been covered exactly once (Koehn et al., 2003).

The search graph of a phrase-based decoder can be interpreted as a lattice, which can be interpreted as a finite state acceptor over target strings. In its most natural form, such an acceptor emits target phrases on each edge, but it can easily be transformed into a form with one edge per token, as shown in Figure 1a. This is sometimes referred to as a word graph (Ueffing et al., 2002), although in our case the segmented phrase table also produces tokens that correspond to morphemes.

Our goal is to desegment the decoder's output lattice, and in doing so, gain access to a compact, desegmented view of a large portion of the translation search space. This can be accomplished by composing the lattice with a **desegmenting transducer** that consumes morphemes and outputs desegmented words. This transducer must be able to consume every word in our lattice's output vocabulary. We define a word using the following regular expression:

$$\text{[prefix]* [stem] [suffix]* | [prefix]+ [suffix]+} \quad (1)$$

where [prefix], [stem] and [suffix] are non-overlapping sets of morphemes, whose members are easily determined using the segmenter's segment boundary markers.[3] The second disjunct of Equation 1 covers words that have no clear stem, such as the Arabic لـ *lh* "for him", segmented as *l+* "for" *+h* "him". Equation 1 may need to be modified for other languages or segmentation schemes, but our techniques generalize to any definition that can be written as a regular expression.

A desegmenting transducer can be constructed by first encoding our desegmenter as a table that maps morpheme sequences to words. Regardless of whether the original desegmenter was based on concatenation, rules or table-lookup, it can be encoded as a lattice-specific table by applying it to an enumeration of all words found in the lattice. We can then transform that table into a finite state transducer with one path per table entry. Finally, we take the closure of this transducer, so that the resulting machine can transduce any sequence of words. The desegmenting trans-

[3] Throughout this paper, we use "+" to mark morphemes as prefixes or suffixes, as in *w+* or *+h*. In Equation 1 only, we overload "+" as the Kleene cross: $X+ == XX*$.

ducer for our running example is shown in Figure 1b. Note that tokens requiring no desegmentation simply emit themselves. The lattice (Figure 1a) can then be desegmented by composing it with the transducer (1b), producing a desegmented lattice (1c). This is a natural place to introduce features that describe the desegmentation process, such as scores provided by a desegmentation table, which can be incorporated into the desegmenting transducer's edge weights.

We now have a desegmented lattice, but it has not been annotated with an unsegmented (word-level) language model. In order to annotate lattice edges with an n-gram LM, every path coming into a node must end with the same sequence of $(n-1)$ tokens. If this property does not hold, then nodes must be split until it does.[4] This property is maintained by the decoder's recombination rules for the segmented LM, but it is not guaranteed for the desegmented LM. Indeed, the expanded word-level context is one of the main benefits of incorporating a word-level LM. Fortunately, LM annotation as well as any necessary lattice modifications can be performed simultaneously by composing the desegmented lattice with a finite state acceptor encoding the LM (Roark et al., 2011).

In summary, we are given a segmented lattice, which encodes the decoder's translation space as an acceptor over morphemes. We compose this acceptor with a desegmenting transducer, and then with an unsegmented LM acceptor, producing a fully annotated, desegmented lattice. Instead of using a tool kit such as OpenFst (Allauzen et al., 2007), we implement both the desegmenting transducer and the LM acceptor programmatically. This eliminates the need to construct intermediate machines, such as the lattice-specific desegmenter in Figure 1b, and facilitates working with edges annotated with feature vectors as opposed to single weights.

Programmatic Desegmentation

Lattice desegmentation is a non-local lattice transformation. That is, the morphemes forming a word might span several edges, making desegmentation non-trivial. Luong et al. (2010) address this problem by forcing the decoder's phrase table to respect word boundaries, guaranteeing that each desegmentable token sequence is local to an edge. Inspired by the use of non-local features in forest decoding (Huang, 2008), we present an algorithm to find **chains** of edges that correspond to desegmentable token sequences, allowing lattice desegmentation with no phrase-table restrictions. This algorithm can be seen as implicitly constructing a customized desegmenting transducer and composing it with the input lattice on the fly.

Before describing the algorithm, we define some notation. An input morpheme lattice is a triple $\langle n_s, \mathcal{N}, \mathcal{E} \rangle$, where $\mathcal{N}$ is a set of nodes, $\mathcal{E}$ is a set of edges, and $n_s \in \mathcal{N}$ is the start node that begins each path through the lattice. Each edge $e \in \mathcal{E}$ is a 4-tuple $\langle \mathit{from}, \mathit{to}, \mathit{lex}, w \rangle$, where from, $\mathit{to} \in \mathcal{N}$ are head and tail nodes, lex is a single token accepted by this edge, and w is the (potentially vector-valued) edge weight. Tokens are drawn from one of three non-overlapping morpho-syntactic sets: $\mathit{lex} \in \mathit{Prefix} \cup \mathit{Stem} \cup \mathit{Suffix}$, where tokens that do not require desegmentation, such as complete words, punctuation and numbers, are considered to be in Stem. It is also useful to consider the set of all outgoing edges for a node $n.\mathit{out} = \{e \in \mathcal{E} | e.\mathit{from} = n\}$.

With this notation in place, we can define a **chain** c to be a sequence of edges $[e_1 \dots e_l]$ such that for $1 \leq i < l : e_i.\mathit{to} = e_{i+1}.\mathit{from}$. We denote singleton chains with $[e]$, and when unambiguous, we abbreviate longer chains with their start and end node $[e_1.\mathit{from} \rightarrow e_l.\mathit{to}]$. A chain is **valid** if it emits the beginning of a word as defined by the regular expression in Equation 1. A valid chain is **complete** if its edges form an entire word, and if it is part of a path through the lattice that consists only of words. In Figure 1a, the complete chains are $[0 \rightarrow 2]$, $[0 \rightarrow 4]$, $[0 \rightarrow 5]$, and $[2 \rightarrow 3]$. The path restriction on complete chains forces words to be bounded by other words in order to be complete.[5] For example, if we removed the edge $2 \rightarrow 3$ (*AlTfl*) from Figure 1a, then $[0 \rightarrow 2]$ ([*b+ lEbp*]) would cease to be a complete chain, but it would still be a valid chain. Note that in the finite-state analogy, the path restriction is implicit in the composition operation.

Algorithm 1 desegments a lattice by finding all complete chains and replacing each one with a single edge. It maintains a work list of nodes that lie on the boundary between words, and for each node on this list, it launches a depth first search

[4] Or the LM composition can be done dynamically, effectively decoding the lattice with a beam or cube-pruned search (Huang and Chiang, 2007).

[5] Sentence-initial suffix morphemes and sentence-final prefix morphemes represent a special case that we omit for the sake of brevity. Lacking stems, they are left segmented.

Algorithm 1 Desegment a lattice $\langle n_s, \mathcal{N}, \mathcal{E} \rangle$

```
{Initialize output lattice and work list WL}
n'_s = n_s, N' = ∅, E' = ∅, WL = [n_s]
while n = WL.pop() do
  {Work on each node only once}
  if n ∈ N' then continue
  N' = N' ∪ {n}
  {Initialize the chain stack C}
  C = ∅
  for e ∈ n.out do
    if [e] is valid then C.push([e])
  {Depth-first search for complete chains}
  while [e_1, ..., e_l] = C.pop() do
    {Attempt to extend chain}
    for e ∈ e_l.to.out do
      if [e_1 ... e_l, e] is valid then
        C.push([e_1, ..., e_l, e])
      else
        Mark [e_1, ..., e_l] as complete
    {Desegment complete chains}
    if [e_1, ..., e_l] is complete then
      WL.push(e_l.to)
      E' = E' ∪ {deseg([e_1, ..., e_l])}
return ⟨n'_s, N', E'⟩
```

to find all complete chains extending from it. The search recognizes the valid chain c to be complete by finding an edge e such that $c + e$ forms a chain, but not a valid one. By inspection of Equation 1, this can only happen when a prefix or stem follows a stem or suffix, which always marks a word boundary. The chains found by this search are desegmented and then added to the output lattice as edges. The nodes at end points of these chains are added to the work list, as they lie at word boundaries by definition. Note that although this algorithm creates completely new edges, the resulting node set $\mathcal{N}'$ will be a subset of the input node set $\mathcal{N}$. The complement $\mathcal{N} - \mathcal{N}'$ will consist of nodes that are word-internal in all paths through the input lattice, such as node 1 in Figure 1a.

Programmatic LM Integration

Programmatic composition of a lattice with an n-gram LM acceptor is a well understood problem. We use a dynamic program to enumerate all $(n-1)$-word contexts leading into a node, and then split the node into multiple copies, one for each context. With each node corresponding to a single LM context, annotation of outgoing edges with n-gram LM scores is straightforward.

3.4 Desegmentation Features

Our re-ranker has access to all of the features used by the decoder, in addition to a number of features enabled by desegmentation.

Desegmentation Score We use a table-based desegmentation method for Arabic, which is based on segmenting an Arabic training corpus and memorizing the observed transformations to reverse them later. Finnish does not require a table, as all words can be desegmented with simple concatenation. The Arabic table consists of $X \rightarrow Y$ entries, where X is a target morpheme sequence and Y is a desegmented surface form. Several entries may share the same X, resulting in multiple desegmentation options. For the sake of symmetry with the unambiguous Finnish case, we augment Arabic n-best lists or lattices with only the most frequent desegmentation Y.[6] We provide the desegmentation score $\log p(Y|X)$= $\log\left(\frac{\text{count of } X \rightarrow Y}{\text{count of X}}\right)$ as a feature, to indicate the entry's ambiguity in the training data.[7] When an X is missing from the table, we fall back on a set of desegmentation rules (El Kholy and Habash, 2012a) and this feature is set to 0. This feature is always 0 for English-Finnish.

Contiguity One advantage of our approach is that it allows discontiguous source words to translate into a single target word. In order to maintain some control over this powerful capability, we create three binary features that indicate the contiguity of a desegmentation. The first feature indicates that the desegmented morphemes were translated from contiguous source words. The second indicates that the source words contained a single discontiguity, as in a word-by-word translation of the "with his blue car" example from Section 2.2. The third indicates two or more discontiguities.

Unsegmented LM A 5-gram LM trained on unsegmented target text is used to assess the fluency of the desegmented word sequence.

4 Experimental Setup

We train our English-to-Arabic system using 1.49 million sentence pairs drawn from the NIST 2012 training set, excluding the UN data. This training set contains about 40 million Arabic tokens before

[6] Allowing the re-ranker to choose between multiple Ys is a natural avenue for future work.

[7] We also experimented on $\log p(X|Y)$ as an additional feature, but observed no improvement in translation quality.

segmentation, and 47 million after segmentation. We tune on the NIST 2004 evaluation set (1353 sentences) and evaluate on NIST 2005 (1056 sentences). As these evaluation sets are intended for Arabic-to-English translation, we select the first English reference to use as our source text.

Our English-to-Finnish system is trained on the same Europarl corpus as Luong et al. (2010) and Clifton and Sarkar (2011), which has roughly one million sentence pairs. We also use their development and test sets (2000 sentences each).

4.1 Segmentation

For Arabic, morphological segmentation is performed by MADA 3.2 (Habash et al., 2009), using the Penn Arabic Treebank (PATB) segmentation scheme as recommended by El Kholy and Habash (2012a). For both segmented and unsegmented Arabic, we further normalize the script by converting different forms of Alif ا آ أ إ and Ya ى ي to bare Alif ا and dotless Ya ى. To generate the desegmentation table, we analyze the segmentations from the Arabic side of the parallel training data to collect mappings from morpheme sequences to surface forms.

For Finnish, we adopt the Unsup L-match segmentation technique of Clifton and Sarkar (2011), which uses Morfessor (Creutz and Lagus, 2005) to analyze the 5,000 most frequent Finnish words. The analysis is then applied to the Finnish side of the parallel text, and a list of segmented suffixes is collected. To improve coverage, words are further segmented according to their longest matching suffix from the list. As Morfessor does not perform any orthographic normalizations, it can be desegmented with simple concatenation.

4.2 Systems

We align the parallel data with GIZA++ (Och et al., 2003) and decode using Moses (Koehn et al., 2007). The decoder's log-linear model includes a standard feature set. Four translation model features encode phrase translation probabilities and lexical scores in both directions. Seven distortion features encode a standard distortion penalty as well as a bidirectional lexicalized reordering model. A KN-smoothed 5-gram language model is trained on the target side of the parallel data with SRILM (Stolcke, 2002). Finally, we include word and phrase penalties. The decoder uses the default parameters for English-to-Arabic, except that the maximum phrase length is set to 8. For English-to-Finnish, we follow Clifton and Sarkar (2011) in setting the hypothesis stack size to 100, distortion limit to 6, and maximum phrase length to 20.

The decoder's log-linear model is tuned with MERT (Och, 2003). Re-ranking models are tuned using a batch variant of hope-fear MIRA (Chiang et al., 2008; Cherry and Foster, 2012), using the n-best variant for n-best desegmentation, and the lattice variant for lattice desegmentation. MIRA was selected over MERT because we have an in-house implementation that can tune on lattices very quickly. During development, we confirmed that MERT and MIRA perform similarly, as is expected with fewer than 20 features. Both the decoder's log-linear model and the re-ranking models are trained on the same development set. Historically, we have not seen improvements from using different tuning sets for decoding and re-ranking. Lattices are pruned to a density of 50 edges per word before re-ranking.

We test four different systems. Our first baseline is **Unsegmented**, where we train on unsegmented target text, requiring no desegmentation step. Our second baseline is **1-best Deseg**, where we train on segmented target text and desegment the decoder's 1-best output. Starting from the system that produced 1-best Deseg, we then output either 1000-best lists or lattices to create our two experimental systems. The **1000-best Deseg** system desegments, augments and re-ranks the decoder's 1000-best list, while **Lattice Deseg** does the same in the lattice. We augment n-best lists and lattices using the features described in Section 3.4.[8]

We evaluate our system using BLEU (Papineni et al., 2002) and TER (Snover et al., 2006). Following Clark et al. (2011), we report average scores over five random tuning replications to account for optimizer instability. For the baselines, this means 5 runs of decoder tuning. For the desegmenting re-rankers, this means 5 runs of re-ranker tuning, each working on n-best lists or lattices produced by the same (representative) decoder weights. We measure statistical significance using MultEval (Clark et al., 2011), which implements a stratified approximate randomization test to account for multiple tuning replications.

[8] Development experiments on a small-data English-to-Arabic scenario indicated that the Desegmentation Score was not particularly useful, so we exclude it from the main comparison, but include it in the ablation experiments.

5 Results

Tables 1 and 2 report results averaged over 5 tuning replications on English-to-Arabic and English-to-Finnish, respectively. In all scenarios, both 1000-best Deseg and Lattice Deseg significantly outperform the 1-best Deseg baseline ($p < 0.01$).

For English-to-Arabic, 1-best desegmentation results in a 0.7 BLEU point improvement over training on unsegmented Arabic. Moving to lattice desegmentation more than doubles that improvement, resulting in a BLEU score of 34.4 and an improvement of 1.0 BLEU point over 1-best desegmentation. 1000-best desegmentation also works well, resulting in a 0.6 BLEU point improvement over 1-best. Lattice desegmentation is significantly better ($p < 0.01$) than 1000-best desegmentation.

For English-to-Finnish, the Unsup L-match segmentation with 1-best desegmentation does not improve over the unsegmented baseline. The segmentation may be addressing issues with model sparsity, but it is also introducing errors that would have been impossible had words been left unsegmented. In fact, even with our lattice desegmenter providing a boost, we are unable to see a significant improvement over the unsegmented model. As we attempted to replicate the approach of Clifton and Sarkar (2011) exactly by working with their segmented data, this difference is likely due to changes in Moses since the publication of their result. Nonetheless, the 1000-best and lattice desegmenters both produce significant improvements over the 1-best desegmentation baseline, with Lattice Deseg achieving a 1-point improvement in TER. These results match the established state-of-the-art on this data set, but also indicate that there is still room for improvement in identifying the best segmentation strategy for English-to-Finnish translation.

We also tried a similar Morfessor-based segmentation for Arabic, which has an unsegmented test set BLEU of 32.7. As in Finnish, the 1-best desegmentation using Morfessor did not surpass the unsegmented baseline, producing a test BLEU of only 31.4 (not shown in Table 1). Lattice desegmentation was able to boost this to 32.9, slightly above 1-best desegmentation, but well below our best MADA desegmentation result of 34.4. There appears to be a large advantage to using MADA's supervised segmentation in this scenario.

Model	Dev	Test	
	BLEU	BLEU	TER
Unsegmented	24.4	32.7	49.4
1-best Deseg	24.4	33.4	48.6
1000-best Deseg	25.0	34.0	48.0
Lattice Deseg	25.2	34.4	47.7

Table 1: Results for English-to-Arabic translation using MADA's PATB segmentation.

Model	Dev	Test	
	BLEU	BLEU	TER
Unsegmented	15.4	15.1	70.8
1-best Deseg	15.3	14.8	71.9
1000-best Deseg	15.4	15.1	71.5
Lattice Deseg	15.5	15.1	70.9

Table 2: Results for English-to-Finnish translation using unsupervised segmentation.

5.1 Ablation

We conducted an ablation experiment on English-to-Arabic to measure the impact of the various features described in Section 3.4. Table 3 compares different combinations of features using lattice desegmentation. The unsegmented LM alone yields a 0.4 point improvement over the 1-best desegmentation score. Adding contiguity indicators on top of the unsegmented LM results in another 0.6 point improvement. As anticipated, the tuner assigns negative weights to discontiguous cases, encouraging the re-ranker to select a safer translation path when possible. Judging from the output on the NIST 2005 test set, the system uses these discontiguous desegmentations very rarely: only 5% of desegmented tokens align to discontiguous source phrases. Adding the desegmentation score to these two feature groups does not improve performance, confirming the results we observed during development. The desegmentation score would likely be useful in a scenario where we provide multiple desegmentation options to the re-ranker; for now, it indicates only the ambiguity of a fixed choice, and is likely redundant with information provided by the language model.

5.2 Error Analysis

In order to better understand the source of our improvements in the English-to-Arabic scenario, we conducted an extensive manual analysis of the differences between 1-best and lattice deseg-

Features	dev	test
1-best Deseg	24.5	33.4
+ Unsegmented LM	24.9	33.8
+ Contiguity	25.2	34.4
+ Desegmentation Score	25.2	34.3

Table 3: The effect of feature ablation on BLEU score for English-to-Arabic translation with lattice desegmentation.

	Lattice Correct	1-best Correct	Both Incorrect
Clitical	157	71	79
Lexical	61	39	80
Inflectional	37	32	47
Part-of-speech	19	17	11

Table 4: Error analysis for English-to-Arabic translation based on 650 sampled instances.

mentation on our test set. We compared the output of the two systems using the Unix tool *wdiff*, which transforms a solution to the longest-common-subsequence problem into a sequence of multi-word insertions and deletions (Hunt and McIlroy, 1976). We considered adjacent insertion-deletion pairs to be (potentially phrasal) substitutions, and collected them into a file, omitting any unpaired insertions or deletions. We then sampled 650 cases where the two sides of the substitution were deemed to be related, and divided these cases into categories based on how the lattice desegmentation differs from the one-best desegmentation. We consider a phrase to be correct only if it can be found in the reference.

Table 4 breaks down per-phrase accuracy according to four manually-assigned categories: (1) **clitical** – the two systems agree on a stem, but at least one clitic, often a prefix denoting a preposition or determiner, was dropped, added or replaced; (2) **lexical** – a word was changed to a morphologically unrelated word with a similar meaning; (3) **inflectional** – the words have the same stem, but different inflection due to a change in gender, number or verb tense; (4) **part-of-speech** – the two systems agree on the lemma, but have selected different parts of speech.

For each case covering a single phrasal difference, we compare the phrases from each system to the reference. We report the number of instances where each system matched the reference, as well as cases where they were both incorrect. The majority of differences correspond to clitics, whose correction appears to be a major source of the improvements obtained by lattice desegmentation. This category is challenging for the decoder because English prepositions tend to correspond to multiple possible forms when translated into Arabic. It also includes the frequent cases involving the nominal determiner prefix *Al* "the" (left unsegmented by the PATB scheme), and the sentence-initial conjunction *w+* "and". The second most common category is lexical, where the unsegmented LM has drastically altered the choice of translation. The remaining categories show no major advantage for either system.

6 Conclusion

We have explored deeper integration of morphological desegmentation into the statistical machine translation pipeline. We have presented a novel, finite-state-inspired approach to lattice desegmentation, which allows the system to account for a desegmented view of many possible translations, without any modification to the decoder or any restrictions on phrase extraction. When applied to English-to-Arabic translation, lattice desegmentation results in a 1.0 BLEU point improvement over one-best desegmentation, and a 1.7 BLEU point improvement over unsegmented translation. We have also applied our approach to English-to-Finnish translation, and although segmentation in general does not currently help, we are able to show significant improvements over a 1-best desegmentation baseline.

In the future, we plan to explore introducing multiple segmentation options into the lattice, and the application of our method to a full morphological analysis (as opposed to segmentation) of the target language. Eventually, we would like to replace the functionality of factored translation models (Koehn and Hoang, 2007) with lattice transformation and augmentation.

Acknowledgments

Thanks to Ann Clifton for generously providing the data and segmentation for our English-to-Finnish experiments, and to Marine Carpuat and Roland Kuhn for their helpful comments on an earlier draft. This research was supported by the Natural Sciences and Engineering Research Council of Canada.

References

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A general and efficient weighted finite-state transducer library. In *Proceedings of the Ninth International Conference on Implementation and Application of Automata, (CIAA 2007)*, volume 4783 of *Lecture Notes in Computer Science*, pages 11–23. Springer. `http://www.openfst.org`.

Ibrahim Badr, Rabih Zbib, and James Glass. 2008. Segmentation for English-to-Arabic statistical machine translation. In *Proceedings of ACL*, pages 153–156.

Ondřej Bojar. 2007. English-to-Czech factored machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 232–239, Prague, Czech Republic, June.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of HLT-NAACL*, Montreal, Canada, June.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proceedings of EMNLP*, pages 224–233.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of ACL*, pages 176–181.

Ann Clifton and Anoop Sarkar. 2011. Combining morpheme-based machine translation with post-processing morpheme prediction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 32–42, Portland, Oregon, USA, June.

Mathias Creutz and Krista Lagus. 2005. Inducing the morphological lexicon of a natural language from unannotated text. In *In Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning (AKRR05*, pages 106–113.

Ahmed El Kholy and Nizar Habash. 2012a. Orthographic and morphological processing for English—Arabic statistical machine translation. *Machine Translation*, 26(1-2):25–45, March.

Ahmed El Kholy and Nizar Habash. 2012b. Translate, predict or generate: Modeling rich morphology in statistical machine translation. *Proceeding of the Meeting of the European Association for Machine Translation*.

Alexander Fraser, Marion Weller, Aoife Cahill, and Fabienne Cap. 2012. Modeling inflection and word-formation in SMT. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 664–674, Avignon, France, April. Association for Computational Linguistics.

Nizar Habash, Owen Rambow, and Ryan Roth. 2009. Mada+tokan: A toolkit for Arabic tokenization, diacritization, morphological disambiguation, POS tagging, stemming and lemmatization. In Khalid Choukri and Bente Maegaard, editors, *Proceedings of the Second International Conference on Arabic Language Resources and Tools*, Cairo, Egypt, April. The MEDAR Consortium.

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

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*, pages 586–594, Columbus, Ohio, June.

James W. Hunt and M. Douglas McIlroy. 1976. An algorithm for differential file comparison. Technical report, Bell Laboratories, June.

Minwoo Jeong, Kristina Toutanova, Hisami Suzuki, and Chris Quirk. 2010. A discriminative lexicon model for complex morphology. In *The Ninth Conference of the Association for Machine Translation in the Americas*.

Philipp Koehn and Hieu Hoang. 2007. Factored translation models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 868–876, Prague, Czech Republic, June. Association for Computational Linguistics.

Philipp Koehn, Franz Joesef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL*, pages 127–133.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June. Association for Computational Linguistics.

Minh-Thang Luong, Preslav Nakov, and Min-Yen Kan. 2010. A hybrid morpheme-word representation for machine translation of morphologically rich languages. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 148–157, Cambridge, MA, October.

Einat Minkov, Kristina Toutanova, and Hisami Suzuki. 2007. Generating complex morphology for machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 128–135, Prague, Czech Republic, June.

Franz Josef Och, Hermann Ney, Franz Josef, and Och Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29.

Franz Joseph Och. 2003. Minimum error rate training for statistical machine translation. In *Proceedings of ACL*, pages 160–167.

Kemal Oflazer and Ilknur Durgar El-Kahlout. 2007. Exploring different representational units in English-to-Turkish statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 25–32, Prague, Czech Republic, June.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Brian Roark, Richard Sproat, and Izhak Shafran. 2011. Lexicographic semirings for exact automata encoding of sequence models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1–5, Portland, Oregon, USA, June.

Mohammad Salameh, Colin Cherry, and Grzegorz Kondrak. 2013. Reversing morphological tokenization in English-to-Arabic SMT. In *Proceedings of the 2013 NAACL HLT Student Research Workshop*, pages 47–53, Atlanta, Georgia, June.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*.

Andreas Stolcke. 2002. Srilm - an extensible language modeling toolkit. In *Intl. Conf. Spoken Language Processing*, pages 901–904.

Michael Subotin. 2011. An exponential translation model for target language morphology. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 230–238, Portland, Oregon, USA, June.

Kristina Toutanova, Hisami Suzuki, and Achim Ruopp. 2008. Applying morphology generation models to machine translation. In *Proceedings of ACL-08: HLT*, pages 514–522, Columbus, Ohio, June.

Nicola Ueffing, Franz J. Och, and Hermann Ney. 2002. Generation of word graphs in statistical machine translation. In *Proceedings of EMNLP*, pages 156–163, Philadelphia, PA, July.

Bilingually-constrained Phrase Embeddings for Machine Translation

Jiajun Zhang[1], **Shujie Liu**[2], **Mu Li**[2], **Ming Zhou**[2] **and Chengqing Zong**[1]
[1]National Laboratory of Pattern Recognition, CASIA, Beijing, P.R. China
{jjzhang,cqzong}@nlpr.ia.ac.cn
[2]Microsoft Research Asia, Beijing, P.R. China
{shujliu,muli,mingzhou}@microsoft.com

Abstract

We propose Bilingually-constrained Recursive Auto-encoders (BRAE) to learn semantic phrase embeddings (compact vector representations for phrases), which can distinguish the phrases with different semantic meanings. The BRAE is trained in a way that minimizes the semantic distance of translation equivalents and maximizes the semantic distance of non-translation pairs simultaneously. After training, the model learns how to embed each phrase semantically in two languages and also learns how to transform semantic embedding space in one language to the other. We evaluate our proposed method on two end-to-end SMT tasks (phrase table pruning and decoding with phrasal semantic similarities) which need to measure semantic similarity between a source phrase and its translation candidates. Extensive experiments show that the BRAE is remarkably effective in these two tasks.

1 Introduction

Due to the powerful capacity of feature learning and representation, Deep (multi-layer) Neural Networks (DNN) have achieved a great success in speech and image processing (Kavukcuoglu et al., 2010; Krizhevsky et al., 2012; Dahl et al., 2012).

Recently, statistical machine translation (SMT) community has seen a strong interest in adapting and applying DNN to many tasks, such as word alignment (Yang et al., 2013), translation confidence estimation (Mikolov et al., 2010; Liu et al., 2013; Zou et al., 2013), phrase reordering prediction (Li et al., 2013), translation modelling (Auli et al., 2013; Kalchbrenner and Blunsom, 2013) and language modelling (Duh et al., 2013; Vaswani et al., 2013). Most of these works attempt to improve some components in SMT based on *word embedding*, which converts a word into a dense, low dimensional, real-valued vector representation (Bengio et al., 2003; Bengio et al., 2006; Collobert and Weston, 2008; Mikolov et al., 2013).

However, in the conventional (phrase-based) SMT, phrases are the basic translation units. The models using word embeddings as the direct inputs to DNN cannot make full use of the whole syntactic and semantic information of the phrasal translation rules. Therefore, in order to successfully apply DNN to model the whole translation process, such as modelling the decoding process, learning compact vector representations for the basic phrasal translation units is the essential and fundamental work.

In this paper, we explore the phrase embedding, which represents a phrase (sequence of words) with a real-valued vector. In some previous works, phrase embedding has been discussed from different views. Socher et al. (2011) make the phrase embeddings capture the sentiment information. Socher et al. (2013a) enable the phrase embeddings to mainly capture the syntactic knowledge. Li et al. (2013) attempt to encode the reordering pattern in the phrase embeddings. Kalchbrenner and Blunsom (2013) utilize a simple convolution model to generate phrase embeddings from word embeddings. Mikolov et al. (2013) consider a phrase as an indivisible n-gram. Obviously, these methods of learning phrase embeddings either focus on some aspects of the phrase (e.g. reordering pattern), or impose strong assumptions (e.g. bag-of-words or indivisible n-gram). Therefore, these phrase embeddings are not suitable to fully represent the phrasal translation units in SMT due to the lack of semantic meanings of the phrase.

Instead, we focus on learning phrase embeddings from the view of semantic meaning, so that our phrase embedding can fully represent the phrase and best fit the phrase-based SMT. Assuming the phrase is a meaningful composition

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 111–121,
Baltimore, Maryland, USA, June 23-25 2014.

of its internal words, we propose Bilingually-constrained Recursive Auto-encoders (BRAE) to learn semantic phrase embeddings. The core idea behind is that a phrase and its correct translation should share the same semantic meaning. Thus, they can supervise each other to learn their semantic phrase embeddings. Similarly, non-translation pairs should have different semantic meanings, and this information can also be used to guide learning semantic phrase embeddings.

In our method, the standard recursive auto-encoder (RAE) pre-trains the phrase embedding with an unsupervised algorithm by minimizing the reconstruction error (Socher et al., 2010), while the bilingually-constrained model learns to fine-tune the phrase embedding by minimizing the semantic distance between translation equivalents and maximizing the semantic distance between non-translation pairs.

We use an example to explain our model. As illustrated in Fig. 1, the Chinese phrase on the left and the English phrase on the right are translations with each other. If we learn the embedding of the Chinese phrase correctly, we can regard it as the gold representation for the English phrase and use it to guide the process of learning English phrase embedding. In the other direction, the Chinese phrase embedding can be learned in the same way. This procedure can be performed with an co-training style algorithm so as to minimize the semantic distance between the translation equivalents [1]. In this way, the result Chinese and English phrase embeddings will capture the semantics as much as possible. Furthermore, a transformation function between the Chinese and English semantic spaces can be learned as well.

With the learned model, we can accurately measure the semantic similarity between a source phrase and a translation candidate. Accordingly, we evaluate the BRAE model on two end-to-end SMT tasks (phrase table pruning and decoding with phrasal semantic similarities) which need to check whether a translation candidate and the source phrase are in the same meaning. In phrase table pruning, we discard the phrasal translation rules with low semantic similarity. In decoding with phrasal semantic similarities, we apply the semantic similarities of the phrase pairs as new features during decoding to guide translation candidate selection. The experiments show that up to 72% of the phrase table can be discarded without significant decrease on the translation quality, and in decoding with phrasal semantic similarities up to 1.7 BLEU score improvement over the state-of-the-art baseline can be achieved.

[1]For simplicity, we do not show non-translation pairs here.

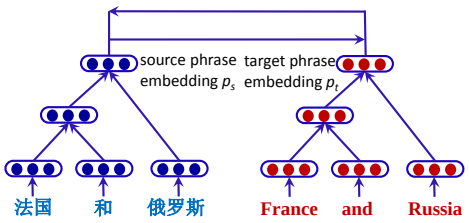


Figure 1: A motivation example for the BRAE model.

In addition, our semantic phrase embeddings have many other potential applications. For instance, the semantic phrase embeddings can be directly fed to DNN to model the decoding process. Besides SMT, the semantic phrase embeddings can be used in other cross-lingual tasks (e.g. cross-lingual question answering) and monolingual applications such as textual entailment, question answering and paraphrase detection.

2 Related Work

Recently, phrase embedding has drawn more and more attention. There are three main perspectives handling this task in monolingual languages.

One method considers the phrases as bag-of-words and employs a convolution model to transform the word embeddings to phrase embeddings (Collobert et al., 2011; Kalchbrenner and Blunsom, 2013). Gao et al. (2013) also use bag-of-words but learn BLEU sensitive phrase embeddings. This kind of approaches does not take the word order into account and loses much information. Instead, our bilingually-constrained recursive auto-encoders not only learn the composition mechanism of generating phrases from words, but also fine tune the word embeddings during the model training stage, so that we can induce the full information of the phrases and internal words.

Another method (Mikolov et al., 2013) deals with the phrases having a meaning that is not a simple composition of the meanings of its individual words, such as *New York Times*. They first find the phrases of this kind. Then, they regard these phrases as indivisible units, and learn their embeddings with the context information. How-

ever, this kind of phrase embedding is hard to capture full semantics since the context of a phrase is limited. Furthermore, this method can only account for a very small part of phrases, since most of the phrases are compositional. In contrast, our method attempts to learn the semantic vector representation for any phrase.

The third method views any phrase as the meaningful composition of its internal words. The recursive auto-encoder is typically adopted to learn the way of composition (Socher et al., 2010; Socher et al., 2011; Socher et al., 2013a; Socher et al., 2013b; Li et al., 2013). They pre-train the RAE with an unsupervised algorithm. And then, they fine-tune the RAE according to the label of the phrase, such as the syntactic category in parsing (Socher et al., 2013a), the polarity in sentiment analysis (Socher et al., 2011; Socher et al., 2013b), and the reordering pattern in SMT (Li et al., 2013). This kind of semi-supervised phrase embedding is in fact performing phrase clustering with respect to the phrase label. For example, in the RAE-based phrase reordering model for SMT (Li et al., 2013), the phrases with the similar reordering tendency (e.g. monotone or swap) are close to each other in the embedding space, such as the prepositional phrases. Obviously, this kind methods of semi-supervised phrase embedding do not fully address the semantic meaning of the phrases. Although we also follow the composition-based phrase embedding, we are the first to focus on the semantic meanings of the phrases and propose a bilingually-constrained model to induce the semantic information and learn transformation of the semantic space in one language to the other.

3 Bilingually-constrained Recursive Auto-encoders

This section introduces the Bilingually-constrained Recursive Auto-encoders (BRAE), that is inspired by two observations. First, the recursive auto-encoder provides a reasonable composition mechanism to embed each phrase. And the semi-supervised phrase embedding (Socher et al., 2011; Socher et al., 2013a; Li et al., 2013) further indicates that phrase embedding can be tuned with respect to the label. Second, even though we have no correct semantic phrase representation as the gold label, the phrases sharing the same meaning provide an indirect but feasible way.

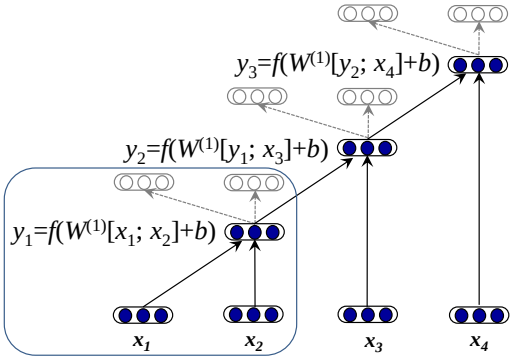


Figure 2: A recursive auto-encoder for a four-word phrase. The empty nodes are the reconstructions of the input.

We will first briefly present the unsupervised phrase embedding, and then describe the semi-supervised framework. After that, we introduce the BRAE on the network structure, objective function and parameter inference.

3.1 Unsupervised Phrase Embedding

3.1.1 Word Vector Representations

In phrase embedding using composition, the word vector representation is the basis and serves as the input to the neural network. After learning word embeddings with DNN (Bengio et al., 2003; Collobert and Weston, 2008; Mikolov et al., 2013), each word in the vocabulary V corresponds to a vector $x \in \mathbb{R}^n$, and all the vectors are stacked into an embedding matrix $L \in \mathbb{R}^{n \times |V|}$.

Given a phrase which is an ordered list of m words, each word has an index i into the columns of the embedding matrix L. The index i is used to retrieve the word's vector representation using a simple multiplication with a binary vector e which is zero in all positions except for the ith index:

$$x_i = Le_i \in \mathbb{R}^n \quad (1)$$

Note that n is usually set empirically, such as $n = 50, 100, 200$. Throughout this paper, $n = 3$ is used for better illustration as shown in Fig. 1.

3.1.2 RAE-based Phrase Embedding

Assuming we are given a phrase $w_1 w_2 \cdots w_m$, it is first projected into a list of vectors $(x_1, x_2, \cdots, x_m)$ using Eq. 1. The RAE learns the vector representation of the phrase by recursively combining two children vectors in a bottom-up manner (Socher et al., 2011). Fig. 2 illustrates an instance of a RAE applied to a binary tree, in

which a standard auto-encoder (in box) is re-used at each node. The standard auto-encoder aims at learning an abstract representation of its input. For two children $c_1 = x_1$ and $c_2 = x_2$, the auto-encoder computes the parent vector y_1 as follows:

$$p = f(W^{(1)}[c_1; c_2] + b^{(1)}) \quad (2)$$

Where we multiply the parameter matrix $W^{(1)} \in \mathbb{R}^{n \times 2n}$ by the concatenation of two children $[c_1; c_2] \in \mathbb{R}^{2n \times 1}$. After adding a bias term $b^{(1)}$, we apply an element-wise activation function such as $f = tanh(\cdot)$, which is used in our experiments. In order to apply this auto-encoder to each pair of children, the representation of the parent p should have the same dimensionality as the c_i's.

To assess how well the parent's vector represents its children, the standard auto-encoder reconstructs the children in a reconstruction layer:

$$[c_1'; c_2'] = f^{(2)}(W^{(2)}p + b^{(2)}) \quad (3)$$

Where c_1' and c_2' are reconstructed children, $W^{(2)}$ and $b^{(2)}$ are parameter matrix and bias term for reconstruction respectively, and $f^{(2)} = tanh(\cdot)$.

To obtain the optimal abstract representation of the inputs, the standard auto-encoder tries to minimize the reconstruction errors between the inputs and the reconstructed ones during training:

$$E_{rec}([c_1; c_2]) = \frac{1}{2}||[c_1; c_2] - [c_1'; c_2']||^2 \quad (4)$$

Given $y_1 = p$, we can use Eq. 2 again to compute y_2 by setting the children to be $[c_1; c_2] = [y_1; x_3]$. The same auto-encoder is re-used until the vector of the whole phrase is generated.

For unsupervised phrase embedding, the only objective is to minimize the sum of reconstruction errors at each node in the optimal binary tree:

$$RAE_\theta(x) = \underset{y \in A(x)}{\operatorname{argmin}} \sum_{s \in y} E_{rec}([c_1; c_2]_s) \quad (5)$$

Where x is the list of vectors of a phrase, and $A(x)$ denotes all the possible binary trees that can be built from inputs x. A greedy algorithm (Socher et al., 2011) is used to generate the optimal binary tree y. The parameters $\theta = (W, b)$ are optimized over all the phrases in the training data.

3.2 Semi-supervised Phrase Embedding

The above RAE is completely unsupervised and can only induce general representations of the multi-word phrases. Several researchers extend the original RAEs to a semi-supervised setting so that the induced phrase embedding can predict a target label, such as polarity in sentiment analysis (Socher et al., 2011), syntactic category in parsing (Socher et al., 2013a) and phrase reordering pattern in SMT (Li et al., 2013).

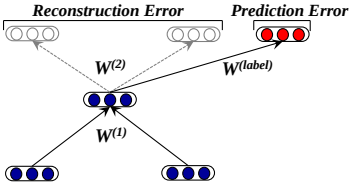


Figure 3: An illustration of a semi-supervised RAE unit. Red nodes show the label distribution.

In the semi-supervised RAE for phrase embedding, the objective function over a (phrase, label) pair (x, t) includes the reconstruction error and the prediction error, as illustrated in Fig. 3.

$$E(x, t; \theta) = \alpha E_{rec}(x, t; \theta) + (1 - \alpha) E_{pred}(x, t; \theta) \quad (6)$$

Where the hyper-parameter α is used to balance the reconstruction and prediction error. For label prediction, the cross-entropy error is usually used to calculate E_{pred}. By optimizing the above objective, the phrases in the vector embedding space will be grouped according to the labels.

3.3 The BRAE Model

We know from the semi-supervised phrase embedding that the learned vector representation can be well adapted to the given label. Therefore, we can imagine that learning semantic phrase embedding is reasonable if we are given gold vector representations of the phrases.

However, no gold semantic phrase embedding exists. Fortunately, we know the fact that the two phrases should share the same semantic representation if they express the same meaning. We can make inference from this fact that if a model can learn the same embedding for any phrase pair sharing the same meaning, the learned embedding must encode the semantics of the phrases and the corresponding model is our desire.

As translation equivalents share the same semantic meaning, we employ high-quality phrase translation pairs as training corpus in this work. Accordingly, we propose the Bilingually-constrained Recursive Auto-encoders (BRAE),

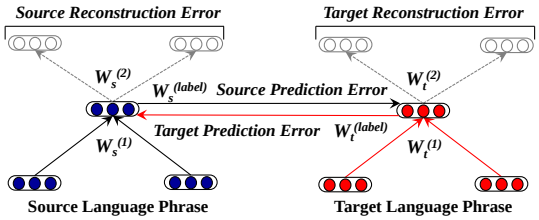


Figure 4: An illustration of the bilingual-constrained recursive auto-encoders. The two phrases are translations with each other.

whose basic goal is to minimize the semantic distance between the phrases and their translations.

3.3.1 The Objective Function

Unlike previous methods, the BRAE model jointly learns two RAEs (Fig. 4 shows the network structure): one for source language and the other for target language. For a phrase pair (s, t), two kinds of errors are involved:

1. **reconstruction error** $E_{rec}(s, t; \theta)$: how well the learned vector representations p_s and p_t represent the phrase s and t respectively?

$$E_{rec}(s, t; \theta) = E_{rec}(s; \theta) + E_{rec}(t; \theta) \quad (7)$$

2. **semantic error** $E_{sem}(s, t; \theta)$: what is the semantic distance between the learned vector representations p_s and p_t?

Since word embeddings for two languages are learned separately and locate in different vector space, we do not enforce the phrase embeddings in two languages to be in the same semantic vector space. We suppose there is a transformation between the two semantic embedding spaces. Thus, the semantic distance is bidirectional: the distance between p_t and the transformation of p_s, and that between p_s and the transformation of p_t. As a result, the overall semantic error becomes:

$$E_{sem}(s, t; \theta) = E_{sem}(s|t, \theta) + E_{sem}(t|s, \theta) \quad (8)$$

Where $E_{sem}(s|t, \theta) = E_{sem}(p_t, f(W_s^l p_s + b_s^l))$ means the transformation of p_s is performed as follows: we first multiply a parameter matrix W_s^l by p_s, and after adding a bias term b_s^l we apply an element-wise activation function $f = tanh(\cdot)$. Finally, we calculate their Euclidean distance:

$$E_{sem}(s|t, \theta) = \frac{1}{2}||p_t - f(W_s^l p_s + b_s^l)||^2 \quad (9)$$

$E_{sem}(t|s, \theta)$ can be calculated in exactly the same way. For the phrase pair (s, t), the joint error is:

$$E(s, t; \theta) = \alpha E_{rec}(s, t; \theta) + (1 - \alpha) E_{sem}(s, t; \theta) \quad (10)$$

The hyper-parameter α weights the reconstruction and semantic error. The final BRAE objective over the phrase pairs training set (S, T) becomes:

$$J_{BRAE} = \frac{1}{N} \sum_{(s,t) \in (S,T)} E(s, t; \theta) + \frac{\lambda}{2}||\theta||^2 \quad (11)$$

3.3.2 Max-Semantic-Margin Error

Ideally, we want the learned BRAE model can make sure that the semantic error for the positive example (a source phrase s and its correct translation t) is much smaller than that for the negative example (the source phrase s and a bad translation t'). However, the current model cannot guarantee this since the above semantic error $E_{sem}(s|t, \theta)$ only accounts for positive ones.

We thus enhance the semantic error with both positive and negative examples, and the corresponding max-semantic-margin error becomes:

$$E^*_{sem}(s|t, \theta) = max\{0, E_{sem}(s|t, \theta) - E_{sem}(s|t', \theta) + 1\} \quad (12)$$

It tries to minimize the semantic distance between translation equivalents and maximize the semantic distance between non-translation pairs simultaneously. Using the above error function, we need to construct a negative example for each positive example. Suppose we are given a positive example (s, t), the correct translation t can be converted into a bad translation t' by replacing the words in t with randomly chosen target language words. Then, a negative example (s, t') is available.

3.3.3 Parameter Inference

Like semi-supervised RAE (Li et al., 2013), the parameters θ in our BRAE model can also be divided into three sets:

θ_L: word embedding matrix L for two languages (Section 3.1.1);

θ_{rec}: recursive auto-encoder parameter matrices $W^{(1)}$, $W^{(2)}$, and bias terms $b^{(1)}$, $b^{(2)}$ for two languages (Section 3.1.2);

θ_{sem}: transformation matrix W^l and bias term b^l for two directions in semantic distance computation (Section 3.3.1).

To have a deep understanding of the parameters, we rewrite Eq. 10:

$$\begin{aligned} E(s,t;\theta) &= \alpha(E_{rec}(s;\theta) + E_{rec}(t;\theta)) \\ &+ (1-\alpha)(E^*_{sem}(s|t,\theta) + E^*_{sem}(t|s,\theta)) \\ &= (\alpha E_{rec}(s;\theta_s) + (1-\alpha)E^*_{sem}(s|t,\theta_s)) \\ &+ (\alpha E_{rec}(t;\theta_t) + (1-\alpha)E^*_{sem}(t|s,\theta_t)) \end{aligned} \quad (13)$$

We can see that the parameters θ can be divided into two classes: θ_s for the source language and θ_t for the target language. The above equation also indicates that the source-side parameters θ_s can be optimized independently as long as the semantic representation p_t of the target phrase t is given to compute $E_{sem}(s|t,\theta)$ with Eq. 9. It is similar for the target-side parameters θ_t.

Assuming the target phrase representation p_t is available, the optimization of the source-side parameters is similar to that of semi-supervised RAE. We apply the Stochastic Gradient Descent (SGD) algorithm to optimize each parameter:

$$\theta_s = \theta_s - \eta \frac{\partial J_s}{\partial \theta_s} \quad (14)$$

In order to run SGD algorithm, we need to solve two problems: one for parameter initialization and the other for partial gradient calculation.

In parameter initialization, θ_{rec} and θ_{sem} for the source language is randomly set according to a normal distribution. For the word embedding L_s, there are two choices. First, L_s is initialized randomly like other parameters. Second, the word embedding matrix L_s is pre-trained with DNN (Bengio et al., 2003; Collobert and Weston, 2008; Mikolov et al., 2013) using large-scale unlabeled monolingual data. We prefer to the second one since this kind of word embedding has already encoded some semantics of the words. In this work, we employ the toolkit Word2Vec (Mikolov et al., 2013) to pre-train the word embedding for the source and target languages. The word embeddings will be fine-tuned in our BRAE model to capture much more semantics.

The partial gradient for one instance is computed as follows:

$$\frac{\partial J_s}{\partial \theta_s} = \frac{\partial E(s|t,\theta_s)}{\partial \theta_s} + \lambda\theta_s \quad (15)$$

Where the source-side error given the target phrase representation includes reconstruction error and updated semantic error:

$$E(s|t,\theta_s) = \alpha E_{rec}(s;\theta_s) + (1-\alpha)E^*_{sem}(s|t,\theta_s) \quad (16)$$

Given the current θ_s, we first construct the binary tree (as illustrated in Fig. 2) for any source-side phrase using the greedy algorithm (Socher et al., 2011). Then, the derivatives for the parameters in the fixed binary tree will be calculated via back-propagation through structures (Goller and Kuchler, 1996). Finally, the parameters will be updated using Eq. 14 and a new θ_s is obtained.

The target-side parameters θ_t can be optimized in the same way as long as the source-side phrase representation p_s is available. It seems a paradox that updating θ_s needs p_t while updating θ_t needs p_s. To solve this problem, we propose an co-training style algorithm which includes three steps:

1. **Pre-training:** applying unsupervised phrase embedding with standard RAE to pre-train the source- and target-side phrase representations p_s and p_t respectively (Section 2.1.2);

2. **Fine-tuning:** with the BRAE model, using target-side phrase representation p_t to update the source-side parameters θ_s and obtain the fine-tuned source-side phrase representation p'_s, and meanwhile using p_s to update θ_t and get the fine-tuned p'_t, and then calculate the joint error over the training corpus;

3. **Termination Check:** if the joint error reaches a local minima or the iterations reach the pre-defined number (25 is used in our experiments), we terminate the training procedure, otherwise we set $p_s = p'_s$, $p_t = p'_t$, and go to step 2.

4 Experiments

With the semantic phrase embeddings and the vector space transformation function, we apply the BRAE to measure the semantic similarity between a source phrase and its translation candidates in the phrase-based SMT. Two tasks are involved in the experiments: phrase table pruning that discards entries whose semantic similarity is very low and decoding with the phrasal semantic similarities as additional new features.

4.1 Hyper-Parameter Settings

The hyper-parameters in the BRAE model include the dimensionality of the word embedding n in Eq. 1, the balance weight α in Eq. 10, λs in Eq. 11, and the learning rate η in Eq. 14.

For the dimensionality n, we have tried three settings $n = 50, 100, 200$ in our experiments. We

empirically set the learning rate $\eta = 0.01$. We draw α from 0.05 to 0.5 with step 0.05, and λs from $\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$. The overall error of the BRAE model is employed to guide the search procedure. Finally, we choose $\alpha = 0.15$, $\lambda_L = 10^{-2}$, $\lambda_{rec} = 10^{-3}$ and $\lambda_{sem} = 10^{-3}$.

4.2 SMT Setup

We have implemented a phrase-based translation system with a maximum entropy based reordering model using the bracketing transduction grammar (Wu, 1997; Xiong et al., 2006).

The SMT evaluation is conducted on Chinese-to-English translation. Accordingly, our BRAE model is trained on Chinese and English. The bilingual training data from LDC [2] contains 0.96M sentence pairs and 1.1M entity pairs with 27.7M Chinese words and 31.9M English words. A 5-gram language model is trained on the Xinhua portion of the English Gigaword corpus and the English part of bilingual training data. The NIST MT03 is used as the development data. NIST MT04-06 and MT08 (news data) are used as the test data. Case-insensitive BLEU is employed as the evaluation metric. The statistical significance test is performed by the re-sampling approach (Koehn, 2004).

In addition, we pre-train the word embedding with toolkit Word2Vec on large-scale monolingual data including the aforementioned data for SMT. The monolingual data contains 1.06B words for Chinese and 1.12B words for English. To obtain high-quality bilingual phrase pairs to train our BRAE model, we perform forced decoding for the bilingual training sentences and collect the phrase pairs used. After removing the duplicates, the remaining 1.12M bilingual phrase pairs (length ranging from 1 to 7) are obtained.

4.3 Phrase Table Pruning

Pruning most of the phrase table without much impact on translation quality is very important for translation especially in environments where memory and time constraints are imposed. Many algorithms have been proposed to deal with this problem, such as significance pruning (Johnson et al., 2007; Tomeh et al., 2009), relevance pruning (Eck et al., 2007) and entropy-based pruning (Ling et al., 2012; Zens et al., 2012). These algorithms are based on corpus statistics including co-occurrence statistics, phrase pair usage and composition information. For example, the significance pruning, which is proven to be a very effective algorithm, computes the probability named p-value, that tests whether a source phrase s and a target phrase t co-occur more frequently in a bilingual corpus than they happen just by chance. The higher the p-value, the more likely of the phrase pair to be spurious.

Our work has the same objective, but instead of using corpus statistics, we attempt to measure the quality of the phrase pair from the view of semantic meaning. Given a phrase pair (s, t), the BRAE model first obtains their semantic phrase representations (p_s, p_t), and then transforms p_s into target semantic space ${p_s}^*$, p_t into source semantic space ${p_t}^*$. We finally get two similarities $Sim({p_s}^*, p_t)$ and $Sim({p_t}^*, p_s)$. Phrase pairs that have a low similarity are more likely to be noise and more prone to be pruned. In experiments, we discard the phrase pair whose similarity in two directions are smaller than a threshold [3].

Table 1 shows the comparison results between our BRAE-based pruning method and the significance pruning algorithm. We can see a common phenomenon in both of the algorithms: for the first few thresholds, the phrase table becomes smaller and smaller while the translation quality is not much decreased, but the performance jumps a lot at a certain threshold (16 for Significance pruning, 0.8 for BRAE-based one).

Specifically, the Significance algorithm can safely discard 64% of the phrase table at its threshold 12 with only 0.1 BLEU loss in the overall test. In contrast, our BRAE-based algorithm can remove 72% of the phrase table at its threshold 0.7 with only 0.06 BLEU loss in the overall evaluation. When the two algorithms using a similar portion of the phrase table [4] (35% in BRAE and 36% in Significance), the BRAE-based algorithm outperforms the Significance algorithm on all the test sets except for MT04. It indicates that our BRAE model is a good alternative for phrase table pruning. Furthermore, our model is much more in-

[2] LDC category numbers: LDC2000T50, LDC2002L27, LDC2003E07, LDC2003E14, LDC2004T07, LDC2005T06, LDC2005T10 and LDC2005T34.

[3] To avoid the situation that all the translation candidates for a source phrase are pruned, we always keep the first 10 best according to the semantic similarity.

[4] In the future, we will compare the performance by enforcing the two algorithms to use the same portion of phrase table

Method	Threshold	PhraseTable	MT03	MT04	MT05	MT06	MT08	ALL
Baseline		100%	35.81	36.91	34.69	33.83	27.17	34.82
BRAE	0.4	52%	35.94	36.96	35.00	34.71	27.77	**35.16**
	0.5	44%	35.67	36.59	34.86	33.91	27.25	**34.89**
	0.6	35%	35.86	36.71	34.93	34.63	27.34	**35.05**
	0.7	28%	35.55	36.62	34.57	33.97	27.10	**34.76**
	0.8	20%	35.06	36.01	34.13	33.04	26.66	34.04
Significance	8	48%	35.86	36.99	34.74	34.53	27.59	**35.13**
	12	36%	35.59	36.73	34.65	34.17	27.16	**34.72**
	16	25%	35.19	36.24	34.26	33.32	26.55	34.09
	20	18%	35.05	36.09	34.02	32.98	26.37	33.97

Table 1: Comparison between BRAE-based pruning and Significance pruning of phrase table. Threshold means similarity in BRAE and negative-log-p-value in Significance. "ALL" combines the development and test sets. **Bold numbers** denote that the result is better than or comparable to that of baseline. $n = 50$ is used for embedding dimensionality.

tuitive because it is directly based on the semantic similarity.

4.4 Decoding with Phrasal Semantic Similarities

Besides using the semantic similarities to prune the phrase table, we also employ them as two informative features like the phrase translation probability to guide translation hypotheses selection during decoding. Typically, four translation probabilities are adopted in the phrase-based SMT, including phrase translation probability and lexical weights in both directions. The phrase translation probability is based on co-occurrence statistics and the lexical weights consider the phrase as bag-of-words. In contrast, our BRAE model focuses on compositional semantics from words to phrases. Therefore, the semantic similarities computed using our BRAE model are complementary to the existing four translation probabilities.

The semantic similarities in two directions $Sim(p_s{}^*, p_t)$ and $Sim(p_t{}^*, p_s)$ are integrated into our baseline phrase-based model. In order to investigate the influence of the dimensionality of the embedding space, we have tried three different settings $n = 50, 100, 200$.

As shown in Table 2, no matter what n is, the BRAE model can significantly improve the translation quality in the overall test data. The largest improvement can be up to 1.7 BLEU score (MT06 for $n = 50$). It is interesting that with dimensionality growing, the translation performance is not consistently improved. We speculate that using $n = 50$ or $n = 100$ can already distinguish good translation candidates from bad ones.

4.5 Analysis on Semantic Phrase Embedding

To have a better intuition about the power of the BRAE model at learning semantic phrase embeddings, we show some examples in Table 3. Given the BRAE model and the phrase training set, we search from the set the most semantically similar English phrases for any new input English phrase.

The input phrases contain different number of words. The table shows that the unsupervised RAE can at most capture the syntactic property when the phrases are short. For example, the unsupervised RAE finds *do not want* for the input phrase *do not agree*. When the phrase becomes longer, the unsupervised RAE cannot even capture the syntactic property. In contrast, our BRAE model learns the semantic meaning for each phrase no matter whether it is short or relatively long. This indicates that the proposed BRAE model is effective at learning semantic phrase embeddings.

5 Discussions

5.1 Applications of The BRAE model

As the semantic phrase embedding can fully represent the phrase, we can go a step further in the phrase-based SMT and feed the semantic phrase embeddings to DNN in order to model the whole translation process (e.g. derivation structure prediction). We will explore this direction in our future work. Besides SMT, the semantic phrase embeddings can be used in other cross-lingual tasks, such as cross-lingual question answering, since the semantic similarity between phrases in different languages can be calculated accurately.

In addition to the cross-lingual applications, we believe the BRAE model can be applied in many

Method	n	MT03	MT04	MT05	MT06	MT08	ALL
Baseline		35.81	36.91	34.69	33.83	27.17	34.82
BRAE	50	36.43	37.64	35.35	35.53	28.59	$\mathbf{35.84}^{+}$
	100	36.45	37.44	35.58	35.42	28.57	$\mathbf{36.03}^{+}$
	200	36.34	37.35	35.78	34.87	27.84	$\mathbf{35.62}^{+}$

Table 2: Experimental results of decoding with phrasal semantic similarities. n is the embedding dimensionality. "+" means that the model significantly outperforms the baseline with $p < 0.01$.

New Phrase	Unsupervised RAE	BRAE
military force	core force main force labor force	military power military strength armed forces
at a meeting	to a meeting at a rate a meeting ,	at the meeting during the meeting at the conference
do not agree	one can accept i can understand do not want	do not favor will not compromise not to approve
each people in this nation	each country regards each country has its each other , and	every citizen in this country all the people in the country people all over the country

Table 3: Semantically similar phrases in the training set for the new phrases.

monolingual NLP tasks which depend on good phrase representations or semantic similarity between phrases, such as named entity recognition, parsing, textual entailment, question answering and paraphrase detection.

5.2 Model Extensions

In fact, the phrases having the same meaning are translation equivalents in different languages, but are paraphrases in one language. Therefore, our model can be easily adapted to learn semantic phrase embeddings using paraphrases.

Our BRAE model still has some limitations. For example, as each node in the recursive auto-encoder shares the same weight matrix, the BRAE model would become weak at learning the semantic representations for long sentences with tens of words. Improving the model to semantically embed sentences is left for our future work.

6 Conclusions and Future Work

This paper has explored the bilingually-constrained recursive auto-encoders in learning phrase embeddings, which can distinguish phrases with different semantic meanings. With the objective to minimize the semantic distance between translation equivalents and maximize the semantic distance between non-translation pairs simultaneously, the learned model can semantically embed any phrase in two languages and can transform the semantic space in one language to the other. Two end-to-end SMT tasks are involved to test the power of the proposed model at learning the semantic phrase embeddings. The experimental results show that the BRAE model is remarkably effective in phrase table pruning and decoding with phrasal semantic similarities.

We have also discussed many other potential applications and extensions of our BRAE model. In the future work, we will explore four directions. 1) we will try to model the decoding process with DNN based on our semantic embeddings of the basic translation units. 2) we are going to learn semantic phrase embeddings with the paraphrase corpus. 3) we will apply the BRAE model in other monolingual and cross-lingual tasks. 4) we plan to learn semantic sentence embeddings by automatically learning different weight matrices for different nodes in the BRAE model.

Acknowledgments

We thank Nan Yang for sharing the baseline code and anonymous reviewers for their valuable comments. The research work has been partially funded by the Natural Science Foundation of China under Grant No. 61333018 and 61303181, and Hi-Tech Research and Development Program (863 Program) of China under Grant No. 2012AA011102.

References

Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint language and translation modeling with recurrent neural networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1044–1054.

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

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167.

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

George E Dahl, Dong Yu, Li Deng, and Alex Acero. 2012. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42.

Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. 2013. Adaptation data selection using neural language models: Experiments in machine translation. In *51st Annual Meeting of the Association for Computational Linguistics*, pages 678–683.

Matthias Eck, Stephen Vogal, and Alex Waibel. 2007. Estimating phrase pair relevance for translation model pruning. In *MTSummit XI*.

Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. 2013. Learning semantic representations for the phrase translation model. *arXiv preprint arXiv:1312.0482*.

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *IEEE International Conference on Neural Networks*, volume 1, pages 347–352.

John Howard Johnson, Joel Martin, George Foster, and Roland Kuhn. 2007. Improving translation quality by discarding most of the phrasetable. In *Proceedings of EMNLP*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709.

Koray Kavukcuoglu, Pierre Sermanet, Y-Lan Boureau, Karol Gregor, Michaël Mathieu, and Yann L Cun. 2010. Learning convolutional feature hierarchies for visual recognition. In *Advances in neural information processing systems*, pages 1090–1098.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP*, pages 388–395.

Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1106–1114.

Peng Li, Yang Liu, and Maosong Sun. 2013. Recursive autoencoders for itg-based translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Wang Ling, Joao Graça, Isabel Trancoso, and Alan Black. 2012. Entropy-based pruning for phrase-based machine translation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 962–971.

Lemao Liu, Taro Watanabe, Eiichiro Sumita, and Tiejun Zhao. 2013. Additive neural networks for statistical machine translation. In *51st Annual Meeting of the Association for Computational Linguistics*, pages 791–801.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048.

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

Richard Socher, Christopher D Manning, and Andrew Y Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013a. Parsing with compositional vector grammars. In *Proceedings of ACL*.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Nadi Tomeh, Nicola Cancedda, and Marc Dymetman. 2009. Complexity-based phrase-table filtering for statistical machine translation. In *Proceedings of Summit XII*, pages 144–151.

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

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

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *Proceedings of ACL-COLING*, pages 505–512.

Nan Yang, Shujie Liu, Mu Li, Ming Zhou, and Nenghai Yu. 2013. Word alignment modeling with context dependent deep neural network. In *51st Annual Meeting of the Association for Computational Linguistics*.

Richard Zens, Daisy Stanton, and Peng Xu. 2012. A systematic comparison of phrase table pruning techniques. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 972–983.

Will Y Zou, Richard Socher, Daniel Cer, and Christopher D Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1393–1398.

Learning New Semi-Supervised Deep Auto-encoder Features for Statistical Machine Translation

Shixiang Lu, Zhenbiao Chen, Bo Xu
Interactive Digital Media Technology Research Center (IDMTech)
Institute of Automation, Chinese Academy of Sciences, Beijing, China
{shixiang.lu,zhenbiao.chen,xubo}@ia.ac.cn

Abstract

In this paper, instead of designing new features based on intuition, linguistic knowledge and domain, we learn some new and effective features using the deep auto-encoder (DAE) paradigm for phrase-based translation model. Using the unsupervised pre-trained deep belief net (DBN) to initialize DAE's parameters and using the input original phrase features as a teacher for semi-supervised fine-tuning, we learn new semi-supervised DAE features, which are more effective and stable than the unsupervised DBN features. Moreover, to learn high dimensional feature representation, we introduce a natural horizontal composition of more DAEs for large hidden layers feature learning. On two Chinese-English tasks, our semi-supervised DAE features obtain statistically significant improvements of 1.34/2.45 (IWSLT) and 0.82/1.52 (NIST) BLEU points over the unsupervised DBN features and the baseline features, respectively.

1 Introduction

Recently, many new features have been explored for SMT and significant performance have been obtained in terms of translation quality, such as syntactic features, sparse features, and reordering features. However, most of these features are manually designed on linguistic phenomena that are related to bilingual language pairs, thus they are very difficult to devise and estimate.

Instead of designing new features based on intuition, linguistic knowledge and domain, for the first time, Maskey and Zhou (2012) explored the possibility of inducing new features in an unsupervised fashion using deep belief net (DBN) (Hinton et al., 2006) for hierarchical phrase-based translation model. Using the 4 original phrase features in the phrase table as the input features, they pre-trained the DBN by contrastive divergence (Hinton, 2002), and generated new unsupervised DBN features using forward computation. These new features are appended as extra features to the phrase table for the translation decoder.

However, the above approach has two major shortcomings. First, the input original features for the DBN feature learning are too simple, the limited 4 phrase features of each phrase pair, such as bidirectional phrase translation probability and bidirectional lexical weighting (Koehn et al., 2003), which are a bottleneck for learning effective feature representation. Second, it only uses the unsupervised layer-wise pre-training of DBN built with stacked sets of Restricted Boltzmann Machines (RBM) (Hinton, 2002), does not have a training objective, so its performance relies on the empirical parameters. Thus, this approach is unstable and the improvement is limited. In this paper, we strive to effectively address the above two shortcomings, and systematically explore the possibility of learning new features using deep (multilayer) neural networks (DNN, which is usually referred under the name *Deep Learning*) for SMT.

To address the first shortcoming, we adapt and extend some simple but effective phrase features as the input features for new DNN feature learning, and these features have been shown significant improvement for SMT, such as, phrase pair similarity (Zhao et al., 2004), phrase frequency, phrase length (Hopkins and May, 2011), and phrase generative probability (Foster et al., 2010), which also show further improvement for new phrase feature learning in our experiments.

To address the second shortcoming, inspired by the successful use of DAEs for handwritten digits recognition (Hinton and Salakhutdinov, 2006; Hinton et al., 2006), information retrieval (Salakhutdinov and Hinton, 2009; Mirowski et

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 122–132,
Baltimore, Maryland, USA, June 23-25 2014.

al., 2010), and speech spectrograms (Deng et al., 2010), we propose new feature learning using semi-supervised DAE for phrase-based translation model. By using the input data as the teacher, the "semi-supervised" fine-tuning process of DAE addresses the problem of "back-propagation without a teacher" (Rumelhart et al., 1986), which makes the DAE learn more powerful and abstract features (Hinton and Salakhutdinov, 2006). For our semi-supervised DAE feature learning task, we use the unsupervised pre-trained DBN to initialize DAE's parameters and use the input original phrase features as the "teacher" for semi-supervised back-propagation. Compared with the unsupervised DBN features, our semi-supervised DAE features are more effective and stable.

Moreover, to learn high dimensional feature representation, we introduce a natural horizontal composition for DAEs (HCDAE) that can be used to create large hidden layer representations simply by horizontally combining two (or more) DAEs (Baldi, 2012), which shows further improvement compared with single DAE in our experiments.

It is encouraging that, non-parametric feature expansion using gaussian mixture model (GMM) (Nguyen et al., 2007), which guarantees invariance to the specific embodiment of the original features, has been proved as a feasible feature generation approach for SMT. Deep models such as DNN have the potential to be much more representationally efficient for feature learning than shallow models like GMM. Thus, instead of GMM, we use DNN (DBN, DAE and HCDAE) to learn new non-parametric features, which has the similar evolution in speech recognition (Dahl et al., 2012; Hinton et al., 2012). DNN features are learned from the non-linear combination of the input original features, they strong capture high-order correlations between the activities of the original features, and we believe this deep learning paradigm induces the original features to further reach their potential for SMT.

Finally, we conduct large-scale experiments on IWSLT and NIST Chinese-English translation tasks, respectively, and the results demonstrate that our solutions solve the two aforementioned shortcomings successfully. Our semi-supervised DAE features significantly outperform the unsupervised DBN features and the baseline features, and our introduced input phrase features significantly improve the performance of DAE feature learning.

The remainder of this paper is organized as follows. Section 2 briefly summarizes the recent related work about the applications of DNN for SMT tasks. Section 3 presents our introduced input features for DNN feature learning. Section 4 describes how to learn our semi-supervised DAE features for SMT. Section 5 describes and discusses the large-scale experimental results. Finally, we end with conclusions in section 6.

2 Related Work

Recently, there has been growing interest in use of DNN for SMT tasks. Le et al. (2012) improved translation quality of n-gram translation model by using a bilingual neural LM, where translation probabilities are estimated using a continuous representation of translation units in lieu of standard discrete representations. Kalchbrenner and Blunsom (2013) introduced recurrent continuous translation models that comprise a class for purely continuous sentence-level translation models. Auli et al. (2013) presented a joint language and translation model based on a recurrent neural network which predicts target words based on an unbounded history of both source and target words. Liu et al. (2013) went beyond the log-linear model for SMT and proposed a novel additive neural networks based translation model, which overcome some of the shortcomings suffered by the log-linear model: linearity and the lack of deep interpretation and representation in features. Li et al. (2013) presented an ITG reordering classifier based on recursive autoencoders, and generated vector space representations for variable-sized phrases, which enable predicting orders to exploit syntactic and semantic information. Lu et al. (2014) adapted and extended the max-margin based RNN (Socher et al., 2011) into HPB translation with force decoding and converting tree, and proposed a RNN based word topology model for HPB translation, which successfully capture the topological structure of the words on the source side in a syntactically and semantically meaningful order.

However, none of these above works have focused on learning new features automatically with input data, and while learning suitable features (representations) is the superiority of DNN since it has been proposed. In this paper, we systematically explore the possibility of learning new fea-

tures using DNN for SMT.

3 Input Features for DNN Feature Learning

The phrase-based translation model (Koehn et al., 2003; Och and Ney, 2004) has demonstrated superior performance and been widely used in current SMT systems, and we employ our implementation on this translation model. Next, we adapt and extend some original phrase features as the input features for DAE feature learning.

3.1 Baseline phrase features

We assume that source phrase $f = f_1, \cdots, f_{l_f}$ and target phrase $e = e_1, \cdots, e_{l_e}$ include l_f and l_e words, respectively. Following (Maskey and Zhou, 2012), we use the following 4 phrase features of each phrase pair (Koehn et al., 2003) in the phrase table as the first type of input features, bidirectional phrase translation probability ($P(e|f)$ and $P(f|e)$), bidirectional lexical weighting ($Lex(e|f)$ and $Lex(f|e)$),

$$X_1 \rightarrow P(f|e), Lex(f|e), P(e|f), Lex(e|f)$$

3.2 Phrase pair similarity

Zhao et al. (2004) proposed a way of using term weight based models in a vector space as additional evidences for phrase pair translation quality. This model employ phrase pair similarity to encode the weights of content and non-content words in phrase translation pairs. Following (Zhao et al., 2004), we calculate bidirectional phrase pair similarity using cosine distance and BM25 distance as,

$$S_i^{cos}(e,f) = \frac{\sum_{j=1}^{l_e}\sum_{i=1}^{l_f} w_{e_j} p(e_j|f_i) w_{f_i}}{sqrt(\sum_{j=1}^{l_e} w_{e_j}^2) sqrt(\sum_{j=1}^{l_e} {w_a^{e_j}}^2)}$$

$$S_d^{cos}(f,e) = \frac{\sum_{i=1}^{l_f}\sum_{j=1}^{l_e} w_{f_i} p(f_i|e_j) w_{e_j}}{sqrt(\sum_{i=1}^{l_f} w_{f_i}^2) sqrt(\sum_{i=1}^{l_f} {w_a^{f_i}}^2)}$$

where, $p(e_j|f_i)$ and $p(f_i|e_j)$ represents bidirectional word translation probability. w_{f_i} and w_{e_j} are term weights for source and target words, $w_a^{e_j}$ and $w_a^{f_i}$ are the transformed weights mapped from all source/target words to the target/source dimension at word e_j and f_i, respectively.

$$S_i^{bm25}(e,f) = \sum_{i=1}^{l_f} idf_{f_i} \frac{(k_1+1) w_{f_i} (k_3+1) w_a^{f_i}}{(K + w_{f_i})(k_3 + w_a^{f_i})}$$

$$S_d^{bm25}(f,e) = \sum_{j=1}^{l_e} idf_{e_j} \frac{(k_1+1) w_{e_j} (k_3+1) w_a^{e_j}}{(K + w_{e_j})(k_3 + w_a^{e_j})}$$

where, k_1, b, k_3 are set to be 1, 1 and 1000, respectively. $K = k_1((1-b) + J/avg(l))$, and J is the phrase length (l_e or l_f), $avg(l)$ is the average phrase length. Thus, we have the second type of input features

$$X_2 \rightarrow S_i^{cos}(f,e), S_i^{bm25}(f,e), S_d^{cos}(e,f), S_d^{bm25}(e,f)$$

3.3 Phrase generative probability

We adapt and extend bidirectional phrase generative probabilities as the input features, which have been used for domain adaptation (Foster et al., 2010). According to the background LMs, we estimate the bidirectional (source/target side) forward and backward phrase generative probabilities as

$$P_f(f) = P(f_1)P(f_2|f_1)\cdots P(f_{l_f}|f_{l_f-n+1}, \cdots, f_{l_f-1})$$
$$P_f(e) = P(e_1)P(e_2|e_1)\cdots P(e_{l_e}|e_{l_e-n+1}, \cdots, e_{l_e-1})$$
$$P_b(f) = P(f_{l_f})P(f_{l_f-1}|f_{l_f})\cdots P(f_1|f_n, \cdots, f_2)$$
$$P_b(e) = P(e_{l_e})P(e_{l_e-1}|e_{l_e})\cdots P(e_1|e_n, \cdots, e_2)$$

where, the bidirectional forward and backward[1] background 4-gram LMs are trained by the corresponding side of bilingual corpus[2]. Then, we have the third type of input features

$$X_3 \rightarrow P_f(e), P_b(e), P_f(f), P_b(f)$$

3.4 Phrase frequency

We consider bidirectional phrase frequency as the input features, and estimate them as

$$P(f) = \frac{count(f)}{\sum_{|f_i|=|f|} count(f_i)}$$

$$P(e) = \frac{count(e)}{\sum_{|e_j|=|e|} count(e_j)}$$

where, the $count(f)$/$count(e)$ are the total number of phrase f/e appearing in the source/target side of the bilingual corpus, and the denominator are the total number of the phrases whose length are equal to $|f|$/$|e|$, respectively. Then, we have the forth type of input features

$$X_4 \rightarrow P(f), P(e)$$

[1] Backward LM has been introduced by Xiong et al. (2011), which successfully capture both the preceding and succeeding contexts of the current word, and we estimate the backward LM by inverting the order in each sentence in the training data from the original order to the reverse order.

[2] This corpus is used to train the translation model in our experiments, and we will describe it in detail in section 5.1.

3.5 Phrase length

Phrase length plays an important role in the translation process (Koehn, 2010; Hopkins and May, 2011). We normalize bidirectional phrase length by the maximum phrase length, and introduce them as the last type of input features

$$X_5 \rightarrow l_e^n, l_f^n$$

In summary, except for the first type of phrase feature X_1 which is used by (Maskey and Zhou, 2012), we introduce another four types of effective phrase features X_2, X_3, X_4 and X_5. Now, the input original phrase features X includes 16 features in our experiments, as follows,

$$X \rightarrow X_1, X_2, X_3, X_4, X_5$$

We build the DAE network where the first layer with visible nodes equaling to 16, and each visible node v_i corresponds to the above original features X in each phrase pair.

4 Semi-Supervised Deep Auto-encoder Features Learning for SMT

Each translation rule in the phrase-based translation model has a set number of features that are combined in the log-linear model (Och and Ney, 2002), and our semi-supervised DAE features can also be combined in this model. In this section, we design our DAE network with various network structures for new feature learning.

4.1 Learning a Deep Belief Net

Inspired by (Maskey and Zhou, 2012), we first learn a deep generative model for feature learning using DBN. DBN is composed of multiple layers of latent variables with the first layer representing the visible feature vectors, which is built with stacked sets of RBMs (Hinton, 2002).

For a RBM, there is full connectivity between layers, but no connections within either layer. The connection weight W, hidden layer biases c and visible layer biases b can be learned efficiently using the contrastive divergence (Hinton, 2002; Carreira-Perpinan and Hinton, 2005). When given a hidden layer h, factorial conditional distribution of visible layer v can be estimated by

$$P(v = 1|h) = \sigma(b + h^T W^T)$$

where σ denotes the logistic sigmoid. Given v, the element-wise conditional distribution of h is

$$P(h = 1|v) = \sigma(c + v^T W)$$

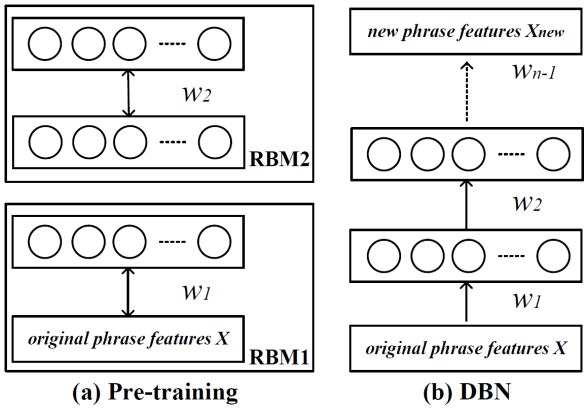


Figure 1: Pre-training consists of learning a stack of RBMs, and these RBMs create an unsupervised DBN.

The two conditional distributions can be shown to correspond to the generative model,

$$P(v, h) = \frac{1}{Z} exp(-E(v, h))$$

where,

$$Z = \sum_{v,h} e^{-E(v,h)}$$

$$E(v, h) = -b^T v - c^T h - v^T W h$$

After learning the first RBM, we treat the activation probabilities of its hidden units, when they are being driven by data, as the data for training a second RBM. Similarly, a n_{th} RBM is built on the output of the $n - 1_{th}$ one and so on until a sufficiently deep architecture is created. These n RBMs can then be composed to form a DBN in which it is easy to infer the states of the n_{th} layer of hidden units from the input in a single forward pass (Hinton et al., 2006), as shown in Figure 1. This greedy, layer-by-layer pre-training can be repeated several times to learn a deep, hierarchical model (DBN) in which each layer of features captures strong high-order correlations between the activities of features in the layer below.

To deal with real-valued input features X in our task, we use an RBM with Gaussian visible units (GRBM) (Dahl et al., 2012) with a variance of 1 on each dimension. Hence, $P(v|h)$ and $E(v, h)$ in the first RBM of DBN need to be modified as

$$P(v|h) = \mathcal{N}(v; b + h^T W^T, I)$$

$$E(v, h) = \frac{1}{2}(v - b)^T (v - b) - c^T h - v^T W h$$

where I is the appropriate identity matrix.

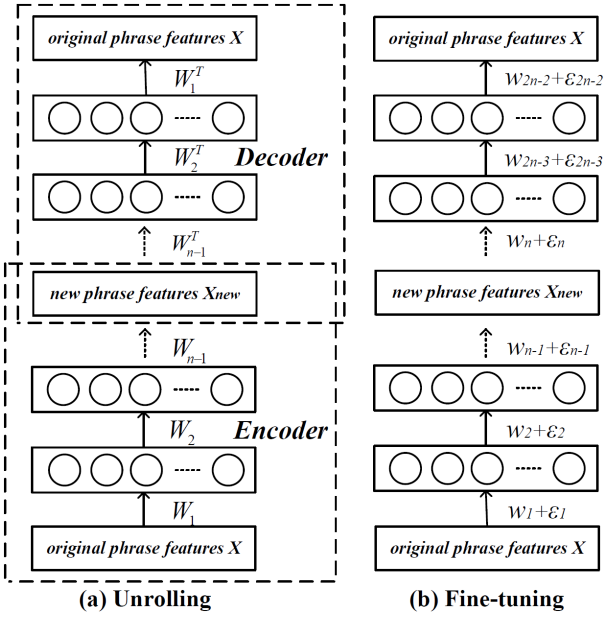


Figure 2: After the unsupervised pre-training, the DBNs are "unrolled" to create a semi-supervised DAE, which is then fine-tuned using back-propagation of error derivatives.

To speed-up the pre-training, we subdivide the entire phrase pairs (with features X) in the phrase table into small mini-batches, each containing 100 cases, and update the weights after each mini-batch. Each layer is greedily pre-trained for 50 epochs through the entire phrase pairs. The weights are updated using a learning rate of 0.1, momentum of 0.9, and a weight decay of 0.0002 $\times$ weight $\times$ learning rate. The weight matrix W are initialized with small random values sampled from a zero-mean normal distribution with variance 0.01.

After the pre-training, for each phrase pair in the phrase table, we generate the DBN features (Maskey and Zhou, 2012) by passing the original phrase features X through the DBN using forward computation.

4.2 From DBN to Deep Auto-encoder

To learn a semi-supervised DAE, we first "unroll" the above n layer DBN by using its weight matrices to create a deep, 2n-1 layer network whose lower layers use the matrices to "encode" the input and whose upper layers use the matrices in reverse order to "decode" the input (Hinton and Salakhutdinov, 2006; Salakhutdinov and Hinton, 2009; Deng et al., 2010), as shown in Figure 2. The layer-wise learning of DBN as above must be treated as a pre-training stage that finds a good region of the parameter space, which is used to initialize our DAE's parameters. Starting in this region, the DAE is then fine-tuned using average squared error (between the output and input) back-propagation to minimize reconstruction error, as to make its output as equal as possible to its input.

For the fine-tuning of DAE, we use the method of conjugate gradients on larger mini-batches of 1000 cases, with three line searches performed for each mini-batch in each epoch. To determine an adequate number of epochs and to avoid over-fitting, we fine-tune on a fraction phrase table and test performance on the remaining validation phrase table, and then repeat fine-tuning on the entire phrase table for 100 epochs.

We experiment with various values for the noise variance and the threshold, as well as the learning rate, momentum, and weight-decay parameters used in the pre-training, the batch size and epochs in the fine-tuning. Our results are fairly robust to variations in these parameters. The precise weights found by the pre-training do not matter as long as it finds a good region of the parameter space from which to start the fine-tuning.

The fine-tuning makes the feature representation in the central layer of the DAE work much better (Salakhutdinov and Hinton, 2009). After the fine-tuning, for each phrase pair in the phrase table, we estimate our DAE features by passing the original phrase features X through the "encoder" part of the DAE using forward computation.

To combine these learned features (DBN and DAE feature) into the log-linear model, we need to eliminate the impact of the non-linear learning mechanism. Following (Maskey and Zhou, 2012), these learned features are normalized by the average of each dimensional respective feature set. Then, we append these features for each phrase pair to the phrase table as extra features.

4.3 Horizontal Composition of Deep Auto-encoders (HCDAE)

Although DAE can learn more powerful and abstract feature representation, the learned features usually have smaller dimensionality compared with the dimensionality of the input features, such as the successful use for handwritten digits recognition (Hinton and Salakhutdinov, 2006; Hinton et al., 2006), information retrieval (Salakhutdinov and Hinton, 2009; Mirowski et al., 2010), and

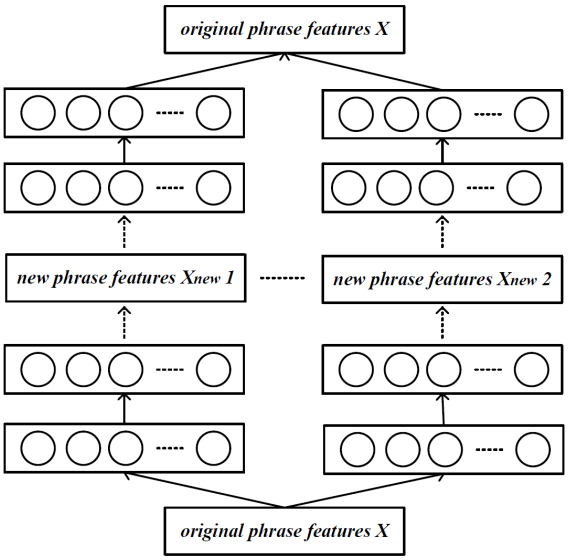


Figure 3: Horizontal composition of DAEs to expand high-dimensional features learning.

4-16-8-2	4-16-8-4	4-16-16-8
4-16-8-4-2	4-16-16-8-4	4-16-16-8-8
4-16-16-8-4-2	4-16-16-8-8-4	4-16-16-16-8-8
4-16-16-8-8-4-2	4-16-16-16-8-8-4	4-16-16-16-16-8-8
6-16-8-2	6-16-8-4	6-16-16-8
6-16-8-4-2	6-16-16-8-4	6-16-16-8-8
6-16-16-8-4-2	6-16-16-8-8-4	6-16-16-16-8-8
6-16-16-16-8-4-2	6-16-16-16-8-8-4	6-16-16-16-16-8-8
8-16-8-2	8-16-8-4	8-16-16-8
8-16-8-4-2	8-16-16-8-4	8-16-16-8-8
8-16-16-8-4-2	8-16-16-8-8-4	8-16-16-16-8-8
8-16-16-16-8-4-2	8-16-16-16-8-8-4	8-16-16-16-16-8-8
16-32-16-2	16-32-16-4	16-32-16-8
16-32-16-8-2	16-32-16-8-4	16-32-32-16-8
16-32-16-8-4-2	16-32-32-16-8-4	16-32-32-16-16-8
16-32-32-16-8-4-2	16-32-32-16-16-8-4	16-32-32-32-16-16-8

Table 1: Details of the used network structure. For example, the architecture 16-32-16-2 (4 layers' network depth) corresponds to the DAE with 16-dimensional input features (X) (input layer), 32/16 hidden units (first/second hidden layer), and 2-dimensional output features (new DAE features) (output layer). During the fine-tuning, the DAE's network structure becomes 16-32-16-2-16-32-16. Correspondingly, 4-16-8-2 and 6(8)-16-8-2 represent the input features are X_1 and X_1+X_i.

speech spectrograms (Deng et al., 2010). Moreover, although we have introduced another four types of phrase features (X_2, X_3, X_4 and X_5), the only 16 features in X are a bottleneck for learning large hidden layers feature representation, because it has limited information, the performance of the high-dimensional DAE features which are directly learned from single DAE is not very satisfactory.

To learn high-dimensional feature representation and to further improve the performance, we introduce a natural horizontal composition for DAEs that can be used to create large hidden layer representations simply by horizontally combining two (or more) DAEs (Baldi, 2012), as shown in Figure 3. Two single DAEs with architectures $16/m_1/16$ and $16/m_2/16$ can be trained and the hidden layers can be combined to yield an expanded hidden feature representation of size $m_1 + m_2$, which can then be fed to the subsequent layers of the overall architecture. Thus, these new $m_1 + m_2$-dimensional DAE features are added as extra features to the phrase table.

Differences in m_1- and m_2-dimensional hidden representations could be introduced by many different mechanisms (e.g., learning algorithms, initializations, training samples, learning rates, or distortion measures) (Baldi, 2012). In our task, we introduce differences by using different initializations and different fractions of the phrase table.

5 Experiments and Results

5.1 Experimental Setup

We now test our DAE features on the following two Chinese-English translation tasks.

IWSLT. The bilingual corpus is the Chinese-English part of Basic Traveling Expression corpus (BTEC) and China-Japan-Korea (CJK) corpus (0.38M sentence pairs with 3.5/3.8M Chinese/English words). The LM corpus is the English side of the parallel data (BTEC, CJK and CWMT08[3]) (1.34M sentences). Our development set is IWSLT 2005 test set (506 sentences), and our test set is IWSLT 2007 test set (489 sentences).

NIST. The bilingual corpus is LDC[4] (3.4M sentence pairs with 64/70M Chinese/English words). The LM corpus is the English side of the parallel data as well as the English Gigaword corpus (LDC2007T07) (11.3M sentences). Our development set is NIST 2005 MT evaluation set (1084 sentences), and our test set is NIST 2006 MT evaluation set (1664 sentences).

We choose the Moses (Koehn et al., 2007) framework to implement our phrase-based machine system. The 4-gram LMs are estimated by the SRILM toolkit with modified Kneser-Ney

[3]the 4th China Workshop on Machine Translation

[4]LDC2002E18, LDC2002T01, LDC2003E07, LDC2003E14, LDC2003T17, LDC2004T07, LDC2004T08, LDC2005T06, LDC2005T10, LDC2005T34, LDC2006T04, LDC2007T09

#		Features	IWSLT		NIST	
			Dev	Test	Dev	Test
1		Baseline	50.81	41.13	36.12	32.59
2	X_1	+DBN_X_1_2f	51.92	42.07*	36.33	33.11*
3		+DAE_X_1_2f	52.49	43.22**	36.92	33.44**
4		+DBN_X_1_4f	51.45	41.78*	36.45	33.12*
5		+DAE_X_1_4f	52.45	43.06**	36.88	33.47**
6		+HCDAE_X_1_2+2f	53.69	43.23***	37.06	33.68***
7		+DBN_X_1_8f	51.74	41.85*	36.61	33.24*
8		+DAE_X_1_8f	52.33	42.98**	36.81	33.36**
9		+HCDAE_X_1_4+4f	52.52	43.26***	37.01	33.63***
10	X	+DBN_X_2f	52.21	42.24*	36.72	33.21*
11		+DAE_X_2f	52.86	43.45**	37.39	33.83**
12		+DBN_X_4f	51.83	42.08*	34.45	33.07*
13		+DAE_X_4f	52.81	43.47**	37.48	33.92**
14		+HCDAE_X_2+2f	53.05	43.58***	37.59	34.11***
15		+DBN_X_8f	51.93	42.01*	36.74	33.29*
16		+DAE_X_8f	52.69	43.26**	37.36	33.75**
17		+HCDAE_X_4+4f	52.93	43.49***	37.53	34.02***
18		+(X_2+X_3+X_4+X_5)	52.23	42.91*	36.96	33.65*
19		+(X_2+X_3+X_4+X_5)+DAE_X_2f	53.55	44.17^{+***}	38.23	34.50^{+***}
20		+(X_2+X_3+X_4+X_5)+DAE_X_4f	53.61	44.22^{+***}	38.28	34.47^{+***}
21		+(X_2+X_3+X_4+X_5)+HCDAE_X_2+2f	53.75	44.28^{+****}	38.35	34.65^{+****}
22		+(X_2+X_3+X_4+X_5)+DAE_X_8f	53.47	44.19^{+***}	38.26	34.46^{+***}
23		+(X_2+X_3+X_4+X_5)+HCDAE_X_4+4f	53.62	44.29^{+****}	38.39	34.57^{+****}

Table 2: The translation results by adding new DNN features (DBN feature (Maskey and Zhou, 2012), our proposed DAE and HCDAE feature) as extra features to the phrase table on two tasks. "DBN_X_1_xf", "DBN_X_xf", "DAE_X_1_xf" and "DAE_X_xf" represent that we use DBN and DAE, input features X_1 and X, to learn x-dimensional features, respectively. "HCDAE_X_x+xf" represents horizontally combining two DAEs and each DAE has the same x-dimensional learned features. All improvements on two test sets are statistically significant by the bootstrap resampling (Koehn, 2004). *: significantly better than the baseline ($p < 0.05$), **: significantly better than "DBN_X_1_xf" or "DBN_X_xf" ($p < 0.01$), ***: significantly better than "DAE_X_1_xf" or "DAE_X_xf" ($p < 0.01$), ****: significantly better than "HCDAE_X_x+xf" ($p < 0.01$), +: significantly better than "X_2+X_3+X_4+X_5" ($p < 0.01$).

discounting. We perform pairwise ranking optimization (Hopkins and May, 2011) to tune feature weights. The translation quality is evaluated by case-insensitive IBM BLEU-4 metric.

The baseline translation models are generated by Moses with default parameter settings. In the contrast experiments, our DAE and HCDAE features are appended as extra features to the phrase table. The details of the used network structure in our experiments are shown in Table 1.

5.2 Results

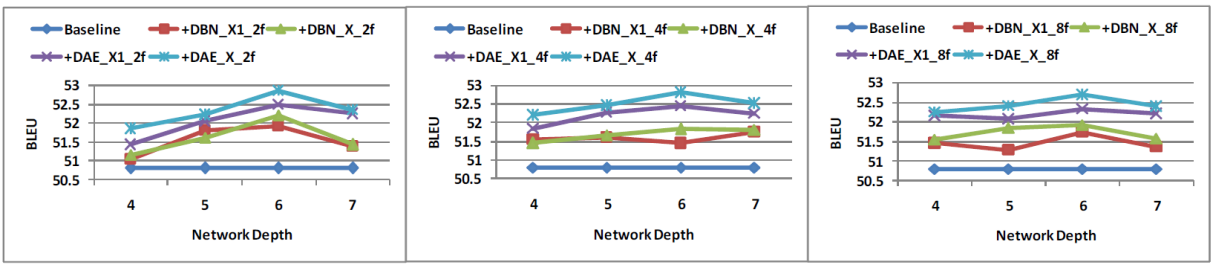
Table 2 presents the main translation results. We use DBN, DAE and HCDAE (with 6 layers' network depth), input features X_1 and X, to learn 2-, 4- and 8-dimensional features, respectively. From the results, we can get some clear trends:

1. Adding new DNN features as extra features significantly improves translation accuracy (row 2-17 vs. 1), with the highest increase of 2.45 (IWSLT) and 1.52 (NIST) (row 14 vs. 1) BLEU points over the baseline features.

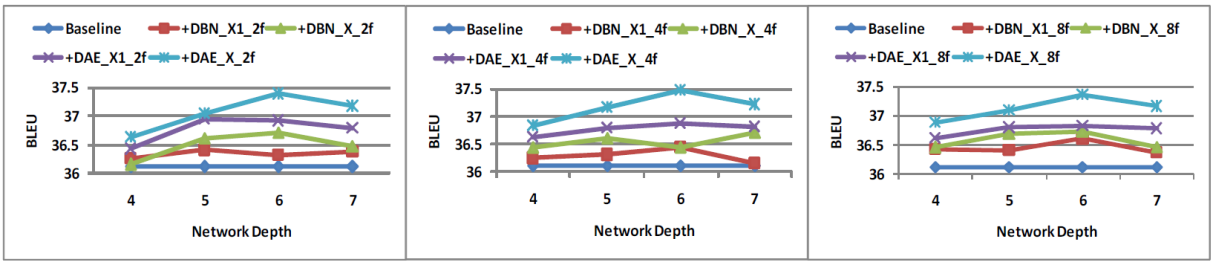
2. Compared with the unsupervised DBN features, our semi-supervised DAE features are more effective for translation decoder (row 3 vs. 2; row 5 vs. 4; row 8 vs. 7; row 11 vs. 10; row 13 vs. 12; row 16 vs. 15). Specially, Table 3 shows the variance distributions of the learned each dimensional DBN and DAE feature, our DAE features have bigger variance distributions which means

Features	IWSLT				NIST			
	σ_1	σ_2	σ_3	σ_4	σ_1	σ_2	σ_3	σ_4
DBN_X_1_4f	0.1678	0.2873	0.2037	0.1622	0.0691	0.1813	0.0828	0.1637
DBN_X_4f	0.2010	0.1590	0.2793	0.1692	0.1267	0.1146	0.2147	0.1051
DAE_X_1_4f	0.5072	0.4486	0.1309	0.6012	0.2136	0.2168	0.2047	0.2526
DAE_X_4f	0.5215	0.4594	0.2371	0.6903	0.2421	0.2694	0.3034	0.2642

Table 3: The variance distributions of each dimensional learned DBN feature and DAE feature on the two tasks.



(a) IWSLT



(b) NIST

Figure 4: The compared results of feature learning with different network structures on two development sets.

Features	IWSLT		NIST	
	Dev	Test	Dev	Test
+DAE_X_1_4f	52.45	43.06	36.88	33.47
+DAE_X_1+X_2_4f	52.76	43.38*	37.28	33.80*
+DAE_X_1+X_3_4f	52.61	43.27*	37.13	33.66*
+DAE_X_1+X_4_4f	52.52	43.24*	36.96	33.58*
+DAE_X_1+X_5_4f	52.49	43.13*	36.96	33.56*
+DAE_X_4f	52.81	43.47*	37.48	33.92*

Table 4: The effectiveness of our introduced input features. "DAE_X_1+X_i_4f" represents that we use DAE, input features $X_1 + X_i$, to learn 4-dimensional features. *: significantly better than "DAE_X_1_4f" ($p < 0.05$).

our DAE features have more discriminative power, and also their variance distributions are more stable.

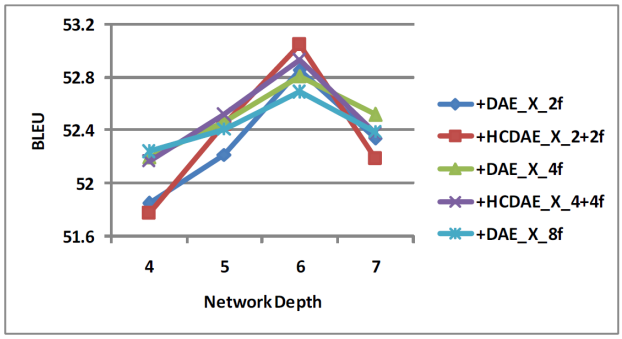
3. HCDAE outperforms single DAE for high dimensional feature learning (row 6 vs. 5; row 9 vs. 8; row 14 vs. 13; row 17 vs. 16), and further improve the performance of DAE feature learning, which can also somewhat address the bring shortcoming of the limited input features.

4. Except for the phrase feature X_1 (Maskey and Zhou, 2012), our introduced input features X significantly improve the DAE feature learning (row 11 vs. 3; row 13 vs. 5; row 16 vs. 8). Specially, Table 4 shows the detailed effectiveness of our introduced input features for DAE feature learning, and the results show that each type of features are very effective for DAE feature learning.

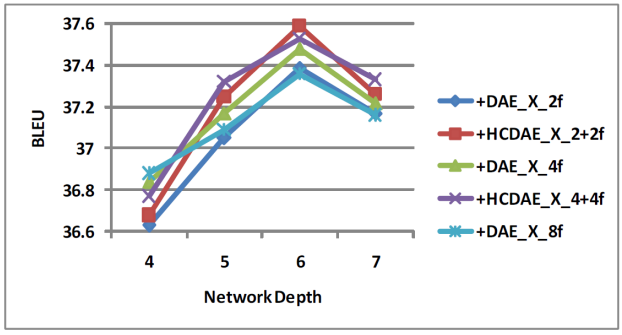
5. Adding the original features (X_2, X_3, X_4 and X_5) and DAE/HCDAE features together can further improve translation performance (row 19-23 vs. 18), with the highest increase of 3.16 (IWSLT) and 2.06 (NIST) (row 21 vs. 1) BLEU points over the baseline features. DAE and HCDAE features are learned from the non-linear combination of the original features, they strong capture high-order correlations between the activities of the original features, as to be further interpreted to reach their potential for SMT. We suspect these learned fea-

tures are complementary to the original features.

5.3 Analysis



(a) IWSLT



(b) NIST

Figure 5: The compared results of using single DAE and the HCDAE for feature learning on two development sets.

Figure 4 shows our DAE features are not only more effective but also more stable than DBN features with various network structures. Also, adding more input features (X vs. X_1) not only significantly improves the performance of DAE feature learning, but also slightly improves the performance of DBN feature learning.

Figure 5 shows there is little change in the performance of using single DAE to learn different dimensional DAE features, but the 4-dimensional features work more better and more stable. HCDAE outperforms the single DAE and learns high-dimensional representation more effectively, especially for the peak point in each condition.

Figures 5 also shows the best network depth for DAE feature learning is 6 layers. When the network depth of DBN is 7 layers, the network depth of corresponding DAE during the fine-tuning is 13 layers. Although we have pre-trained the corresponding DBN, this DAE network is so deep, the fine-tuning does not work very well and typically finds poor local minima. We suspect this leads to the decreased performance.

6 Conclusions

In this paper, instead of designing new features based on intuition, linguistic knowledge and domain, we have learned new features using the DAE for the phrase-based translation model. Using the unsupervised pre-trained DBN to initialize DAE's parameters and using the input original phrase features as the "teacher" for semi-supervised back-propagation, our semi-supervised DAE features are more effective and stable than the unsupervised DBN features (Maskey and Zhou, 2012). Moreover, to further improve the performance, we introduce some simple but effective features as the input features for feature learning. Lastly, to learn high dimensional feature representation, we introduce a natural horizontal composition of two DAEs for large hidden layers feature learning.

On two Chinese-English translation tasks, the results demonstrate that our solutions solve the two aforementioned shortcomings successfully. Firstly, our DAE features obtain statistically significant improvements of 1.34/2.45 (IWSLT) and 0.82/1.52 (NIST) BLEU points over the DBN features and the baseline features, respectively. Secondly, compared with the baseline phrase features X_1, our introduced input original phrase features X significantly improve the performance of not only our DAE features but also the DBN features.

The results also demonstrate that DNN (DAE and HCDAE) features are complementary to the original features for SMT, and adding them together obtain statistically significant improvements of 3.16 (IWSLT) and 2.06 (NIST) BLEU points over the baseline features. Compared with the original features, DNN (DAE and HCDAE) features are learned from the non-linear combination of the original features, they strong capture high-order correlations between the activities of the original features, and we believe this deep learning paradigm induces the original features to further reach their potential for SMT.

Acknowledgments

This work was supported by 863 program in China (No. 2011AA01A207). We would like to thank Xingyuan Peng, Lichun Fan and Hongyan Li for their helpful discussions. We also thank the anonymous reviewers for their insightful comments.

References

Michael Auli, Michel Galley, Chris Quirk and Geoffrey Zweig. 2013. Joint language and translation modeling with recurrent neural networks. In *Proceedings of EMNLP*, pages 1044-1054.

Pierre Baldi. 2012. Autoencoders, unsupervised learning, and deep architectures. *JMLR: workshop on unsupervised and transfer learning*, 27:37-50.

Miguel A. Carreira-Perpinan and Geoffrey E. Hinton. 2005. On contrastive divergence learning. In *Proceedings of AI and Statistics*.

George Dahl, Dong Yu, Li Deng, and Alex Acero. 2012. Context-dependent pre-trained deep neural networks for large vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30-42.

Li Deng, Mike Seltzer, Dong Yu, Alex Acero, Abdelrahman Mohamed, and Geoffrey E. Hinton. 2010. Binary coding of speech spectrograms using a deep auto-encoder. In *Proceedings of INTERSPEECH*, pages 1692-1695.

George Foster, Cyril Goutte, and Roland Kuhn. 2010. Discriminative instance weighting for domain adaptation in statistical machine translation. In *Proceedings of EMNLP*, pages 451-459.

Geoffrey E. Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771-1800.

Geoffrey E. Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury. 2012. Deep neural networks for acoustic modeling in speech tecognition. *IEEE Signal Processing Magazine*, 29(6):82-97.

Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. 2001. Transforming auto-encoders. In *Proceedings of ANN*.

Geoffrey E. Hinton and Ruslan R. Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science*, 313:504-507.

Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527-1544.

Mark Hopkins and Jonathan May 2011. Tuning as ranking. In *Proceedings of EMNLP*, pages 1352-1362.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of EMNLP*, pages 1700-1709.

Philipp Koehn. 2004. Statistical significance tests from achine translation evaluation. In *Proceedings of ACL*, pages 388-395.

Philipp Koehn. 2010. Statistical machine translation. *Cambridge University Press*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL, Demonstration Session*, pages 177-180.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL*, pages 48-54.

Hai-Son Le, Alexandre Allauzen, and François Yvon. 2012. Continuous space translation models with neural networks. In *Proceedings of NAACL*, pages 39-48.

Peng Li, Yang Liu, Maosong Sun. 2013. Recursive autoencoders for ITG-based translation. In *Proceedings of EMNLP*, pages 567-577.

Lemao Liu, Taro Watanabe, Eiichiro Sumita, and Tiejun Zhao. 2013. Additive neural networks for statistical machine translation. In *Proceedings of ACL*, pages 791-801.

Shixiang Lu, Wei Wei, Xiaoyin Fu and Bo Xu. 2014. Recursive neural network based word topology model for hierarchical phrase-based speech translation. In *Proceedings of ICASSP*.

Yuval Marton and Philip Resnik. 2008. Soft syntactic constraints for hierarchical phrase-based translation. In *Proceedings of ACL*, pages 1003-1011.

Sameer Maskey and Bowen Zhou. 2012. Unsupervised deep belief features for speech translation. In *Proceedings of INTERSPEECH*.

Piotr Mirowski, MarcAurelio Ranzato, and Yann LeCun. 2010. Dynamic auto-encoders for semantic indexing. In *Proceedings of NIPS-2010 Workshop on Deep Learning*.

Patrick Nguyen, Milind Mahajan, and Xiaodong He. 2007. Training non-parametric features for statistical machine translation. In *Proceedings of WMT*, pages 72-79.

Franz J. Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of ACL*, pages 440-447.

Franz J. Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of ACL*, pages 295-302.

Franz J. Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417-449.

David Rumelhart, Geoffrey E. Hinton, and Ronale Williams. 1986. Learning internal representations by back-propagation errors. *Parallel Distributed Processing, Vol 1: Foundations, MIT Press*.

Ruslan R. Salakhutdinov and Geoffrey E. Hinton. 2009. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969-978.

Richard Socher, Cliff C. Lin, Andrew Y. Ng, and Christopher D. Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of ICML*.

Deyi Xiong, Min Zhang, and Haizhou Li. 2011. Enhancing language models in statistical machine translation with backward n-grams and mutual information triggers. In *Proceedings of ACL*, pages 1288-1297.

Bing Zhao, Stephan Vogel, and Alex Waibel. 2004. Phrase pair rescoring with term weightings for statistical machine translation. In *Proceedings of EMNLP*.

Learning Topic Representation for SMT with Neural Networks*

Lei Cui[1], **Dongdong Zhang**[2], **Shujie Liu**[2], **Qiming Chen**[3], **Mu Li**[2], **Ming Zhou**[2], and **Muyun Yang**[1]

[1]School of Computer Science and Technology, Harbin Institute of Technology, Harbin, P.R. China
leicui@hit.edu.cn, ymy@mtlab.hit.edu.cn
[2]Microsoft Research, Beijing, P.R. China
{dozhang,shujliu,muli,mingzhou}@microsoft.com
[3]Shanghai Jiao Tong University, Shanghai, P.R. China
simoncqm@gmail.com

Abstract

Statistical Machine Translation (SMT) usually utilizes contextual information to disambiguate translation candidates. However, it is often limited to contexts within sentence boundaries, hence broader topical information cannot be leveraged. In this paper, we propose a novel approach to learning topic representation for parallel data using a neural network architecture, where abundant topical contexts are embedded via topic relevant monolingual data. By associating each translation rule with the topic representation, topic relevant rules are selected according to the distributional similarity with the source text during SMT decoding. Experimental results show that our method significantly improves translation accuracy in the NIST Chinese-to-English translation task compared to a state-of-the-art baseline.

1 Introduction

Making translation decisions is a difficult task in many Statistical Machine Translation (SMT) systems. Current translation modeling approaches usually use context dependent information to disambiguate translation candidates. For example, translation sense disambiguation approaches (Carpuat and Wu, 2005; Carpuat and Wu, 2007) are proposed for phrase-based SMT systems. Meanwhile, for hierarchical phrase-based or syntax-based SMT systems, there is also much work involving rich contexts to guide rule selection (He et al., 2008; Liu et al., 2008; Marton and Resnik, 2008; Xiong et al., 2009). Although these methods are effective and proven successful in many SMT systems, they only leverage within-sentence contexts which are insufficient in exploring broader information. For example, the word *driver* often means "the operator of a motor vehicle" in common texts. But in the sentence "Finally, we write the user response to the buffer, i.e., pass it to our driver", we understand that *driver* means "computer program". In this case, people understand the meaning because of the IT topical context which goes beyond sentence-level analysis and requires more relevant knowledge. Therefore, it is important to leverage topic information to learn smarter translation models and achieve better translation performance.

Topic modeling is a useful mechanism for discovering and characterizing various semantic concepts embedded in a collection of documents. Attempts on topic-based translation modeling include topic-specific lexicon translation models (Zhao and Xing, 2006; Zhao and Xing, 2007), topic similarity models for synchronous rules (Xiao et al., 2012), and document-level translation with topic coherence (Xiong and Zhang, 2013). In addition, topic-based approaches have been used in domain adaptation for SMT (Tam et al., 2007; Su et al., 2012), where they view different topics as different domains. One typical property of these approaches in common is that they only utilize parallel data where document boundaries are explicitly given. In this way, the topic of a sentence can be inferred with document-level information using off-the-shelf topic modeling toolkits such as Latent Dirichlet Allocation (LDA) (Blei et al., 2003) or Hidden Topic Markov Model (HTMM) (Gruber et al., 2007). Most of them also assume that the input must be in document level. However, this situation does not always happen since there is considerable amount of parallel data which does not have document boundaries. In addition, contemporary SMT systems often works on sentence level rather than document level due to the efficiency. Although we can easily apply LDA at the

*This work was done while the first and fourth authors were visiting Microsoft Research.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 133–143,
Baltimore, Maryland, USA, June 23-25 2014.

sentence level, it is quite difficult to infer the topic accurately with only a few words in the sentence. This makes previous approaches inefficient when applied them in real-world commercial SMT systems. Therefore, we need to devise a systematical approach to enriching the sentence and inferring its topic more accurately.

In this paper, we propose a novel approach to learning topic representations for sentences. Since the information within the sentence is insufficient for topic modeling, we first enrich sentence contexts via Information Retrieval (IR) methods using content words in the sentence as queries, so that topic-related monolingual documents can be collected. These topic-related documents are utilized to learn a specific topic representation for each sentence using a neural network based approach. Neural network is an effective technique for learning different levels of data representations. The levels inferred from neural network correspond to distinct levels of concepts, where high-level representations are obtained from low-level bag-of-words input. It is able to detect correlations among any subset of input features through non-linear transformations, which demonstrates the superiority of eliminating the effect of noisy words which are irrelevant to the topic. Our problem fits well into the neural network framework and we expect that it can further improve inferring the topic representations for sentences.

To incorporate topic representations as translation knowledge into SMT, our neural network based approach directly optimizes similarities between the source language and target language in a compact topic space. This underlying topic space is learned from sentence-level parallel data in order to share topic information across the source and target languages as much as possible. Additionally, our model can be discriminatively trained with a large number of training instances, without expensive sampling methods such as in LDA or HTMM, thus it is more practicable and scalable. Finally, we associate the learned representation to each bilingual translation rule. Topic-related rules are selected according to distributional similarity with the source text, which helps hypotheses generation in SMT decoding. We integrate topic similarity features in the log-linear model and evaluate the performance on the NIST Chinese-to-English translation task. Experimental results demonstrate that our model significantly improves translation accuracy over a state-of-the-art baseline.

2 Background: Deep Learning

Deep learning is an active topic in recent years which has triumphed in many machine learning research areas. This technique began raising public awareness in the mid-2000s after researchers showed how a multi-layer feed-forward neural network can be effectively trained. The training procedure often involves two phases: a layer-wise unsupervised pre-training phase and a supervised fine-tuning phase. For pre-training, Restricted Boltzmann Machine (RBM) (Hinton et al., 2006), auto-encoding (Bengio et al., 2006) and sparse coding (Lee et al., 2006) are most frequently used. Unsupervised pre-training trains the network one layer at a time and helps guide the parameters of the layer towards better regions in parameter space (Bengio, 2009). Followed by fine-tuning in this parameter region, deep learning is able to achieve state-of-the-art performance in various research areas, including breakthrough results on the ImageNet dataset for objective recognition (Krizhevsky et al., 2012), significant error reduction in speech recognition (Dahl et al., 2012), etc.

Deep learning has also been successfully applied in a variety of NLP tasks such as part-of-speech tagging, chunking, named entity recognition, semantic role labeling (Collobert et al., 2011), parsing (Socher et al., 2011a), sentiment analysis (Socher et al., 2011b), etc. Most NLP research converts a high-dimensional and sparse binary representation into a low-dimensional and real-valued representation. This low-dimensional representation is usually learned from huge amount of monolingual texts in the pre-training phase, and then fine-tuned towards task-specific criterion. Inspired by previous successful research, we first learn sentence representations using topic-related monolingual texts in the pre-training phase, and then optimize the bilingual similarity by leveraging sentence-level parallel data in the fine-tuning phase.

3 Topic Similarity Model with Neural Network

In this section, we explain our neural network based topic similarity model in detail, as well as how to incorporate the topic similarity features into SMT decoding procedure. Figure 1 sketches the high-level overview which illustrates how to

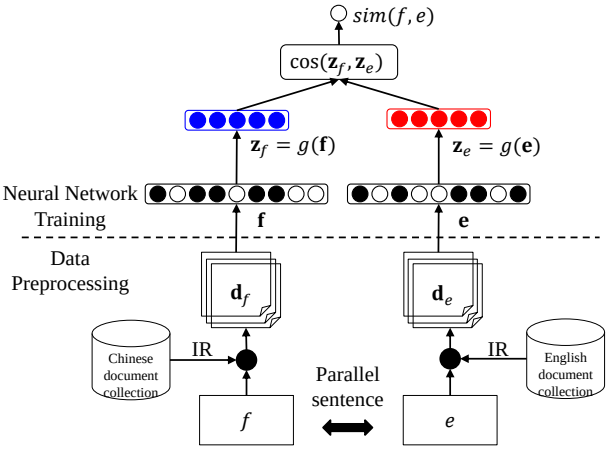


Figure 1: Overview of neural network based topic similarity model.

learn topic representations using sentence-level parallel data. Given a parallel sentence pair $\langle f, e \rangle$, the first step is to treat f and e as queries, and use IR methods to retrieve relevant documents to enrich contextual information for them. Specifically, the ranking model we used is a Vector Space Model (VSM), where the query and document are converted into tf-idf weighted vectors. The most relevant N documents $\mathbf{d}_f$ and $\mathbf{d}_e$ are retrieved and converted to a high-dimensional, bag-of-words input $\mathbf{f}$ and $\mathbf{e}$ for the representation learning[1].

There are two phases in our neural network training process: pre-training and fine-tuning. In the pre-training phase (Section 3.1), we build two neural networks with the same structure but different parameters to learn a low-dimensional representation for sentences in two different languages. Then, in the fine-tuning phase (Section 3.2), our model directly optimizes the similarity of two low-dimensional representations, so that it highly correlates to SMT decoding. Finally, the learned representation is used to calculate similarities which are integrated as features in SMT decoding procedure (Section 3.3).

3.1 Pre-training using denoising auto-encoder

In the pre-training phase, we leverage neural network structures to transform high-dimensional sparse vectors to low-dimensional dense vectors. The topic similarity is calculated on top of the learned dense vectors. This dense representation should preserve the information from the bag-of-words input, meanwhile alleviate data sparse problem. Therefore, we use a specially designed mechanism called auto-encoder to solve this problem. Auto-encoder (Bengio et al., 2006) is one of the basic building blocks of deep learning. Assuming that the input is a n-of-V binary vector $\mathbf{x}$ representing the bag-of-words (V is the vocabulary size), an auto-encoder consists of an encoding process $g(\mathbf{x})$ and a decoding process $h(g(\mathbf{x}))$. The objective of the auto-encoder is to minimize the reconstruction error $\mathcal{L}(h(g(\mathbf{x})), \mathbf{x})$. Our goal is to learn a low-dimensional vector which can preserve information from the original n-of-V vector.

One problem with auto-encoder is that it treats all words in the same way, making no distinguishment between function words and content words. The representation learned by auto-encoders tends to be influenced by the function words, thereby it is not robust. To alleviate this problem, Vincent et al. (2008) proposed the Denoising Auto-Encoder (DAE), which aims to reconstruct a clean, "repaired" input from a corrupted, partially destroyed vector. This is done by corrupting the initial input $\mathbf{x}$ to get a partially destroyed version $\tilde{\mathbf{x}}$. DAE is capable of capturing the global structure of the input while ignoring the noise. In our task, for each sentence, we treat the retrieved N relevant documents as a single large document and convert it to a bag-of-words vector $\mathbf{x}$ in Figure 2. With DAE, the input $\mathbf{x}$ is manually corrupted by applying masking noise (randomly mask 1 to 0) and getting $\tilde{\mathbf{x}}$. Denoising training is considered as "filling in the blanks" (Vincent et al., 2010), which means the masking components can be recovered from the non-corrupted components. For example, in IT related texts, if the word *driver* is masked, it should be predicted through hidden units in neural networks by active signals such as "buffer", "user response", etc.

In our case, the encoding process transforms the corrupted input $\tilde{\mathbf{x}}$ into $g(\tilde{\mathbf{x}})$ with two layers: a linear layer connected with a non-linear layer. Assuming that the dimension of the $g(\tilde{\mathbf{x}})$ is L, the linear layer forms a $L \times V$ matrix W which projects the n-of-V vector to a L-dimensional hidden layer. After the bag-of-words input has been transformed, they are fed into a subsequent layer to model the highly non-linear relations among words:

$$\mathbf{z} = f(W\tilde{\mathbf{x}} + \mathbf{b}) \quad (1)$$

where $\mathbf{z}$ is the output of the non-linear layer, $\mathbf{b}$ is a

[1]We use $\mathbf{f}$ and $\mathbf{e}$ to denote the n-of-V vector converted from the retrieved documents.

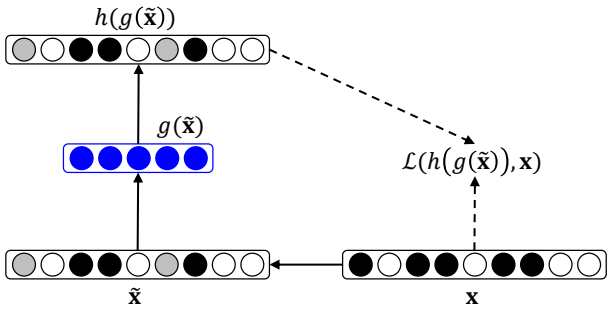


Figure 2: Denoising auto-encoder with a bag-of-words input.

L-length bias vector. $f(\cdot)$ is a non-linear function, where common choices include sigmoid function, hyperbolic function, "hard" hyperbolic function, rectifier function, etc. In this work, we use the rectifier function as our non-linear function due to its efficiency and better performance (Glorot et al., 2011):

$$rec(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

The decoding process consists of a linear layer and a non-linear layer with similar network structures, but different parameters. It transforms the L-dimensional vector $g(\tilde{\mathbf{x}})$ to a V-dimensional vector $h(g(\tilde{\mathbf{x}}))$. To minimize reconstruction error with respect to $\tilde{\mathbf{x}}$, we define the loss function as the L2-norm of the difference between the uncorrupted input and reconstructed input:

$$\mathcal{L}(h(g(\tilde{\mathbf{x}})), \mathbf{x}) = \|h(g(\tilde{\mathbf{x}})) - \mathbf{x}\|_2 \quad (3)$$

Multi-layer neural networks are trained with the standard back-propagation algorithm (Rumelhart et al., 1988). The gradient of the loss function is calculated and back-propagated to the previous layer to update its parameters. Training neural networks involves many factors such as the learning rate and the length of hidden layers. We will discuss the optimization of these parameters in Section 4.

3.2 Fine-tuning with parallel data

In the fine-tuning phase, we stack another layer on top of the two low-dimensional vectors to maximize the similarity between source and target languages. The similarity scores are integrated into the standard log-linear model for making translation decisions. Since the vectors from DAE are trained using information from monolingual training data independently, these vectors may be inadequate to measure bilingual topic similarity due to their different topic spaces. Therefore, in this stage, parallel sentence pairs are used to help connecting the vectors from different languages because they express the same topic. In fact, the objective of fine-tuning is to discover a latent topic space which is shared by both languages as much as possible. This shared topic space is particularly useful when the SMT decoder tries to match the source texts and translation candidates in the target language.

Given a parallel sentence pair $\langle f, e \rangle$, the DAE learns representations for f and e respectively, as $\mathbf{z}_f = g(\mathbf{f})$ and $\mathbf{z}_e = g(\mathbf{e})$ in Figure 1. We then take two vectors as the input to calculate their similarity. Consequently, the whole neural network can be fine-tuned towards the supervised criteria with the help of parallel data. The similarity score of the representation pair $\langle \mathbf{z}_f, \mathbf{z}_e \rangle$ is defined as the cosine similarity of the two vectors:

$$\begin{aligned} sim(f, e) &= \cos(\mathbf{z}_f, \mathbf{z}_e) \\ &= \frac{\mathbf{z}_f \cdot \mathbf{z}_e}{\|\mathbf{z}_f\| \|\mathbf{z}_e\|} \end{aligned} \quad (4)$$

Since a parallel sentence pair should have the same topic, our goal is to maximize the similarity score between the source sentence and target sentence. Inspired by the contrastive estimation method (Smith and Eisner, 2005), for each parallel sentence pair $\langle f, e \rangle$ as a positive instance, we select another sentence pair $\langle f', e' \rangle$ from the training data and treat $\langle f, e' \rangle$ as a negative instance. To make the similarity of the positive instance larger than the negative instance by some margin η, we utilize the following pairwise ranking loss:

$$\mathcal{L}(f, e) = \max\{0, \eta - sim(f, e) + sim(f, e')\} \quad (5)$$

where $\eta = \frac{1}{2} - sim(f, f')$. The rationale behind this criterion is, the smaller $sim(f, f')$ is, the more we should penalize negative instances.

To effectively train the model in this task, negative instances must be selected carefully. Since different sentences may have very similar topic distributions, we select negative instances that are dissimilar with the positive instances based on the following criteria:

1. For each positive instance $\langle f, e \rangle$, we select e' which contains at least 30% different content words from e.

2. If we cannot find such e', remove $\langle f, e\rangle$ from the training instances for network learning.

The model minimizes the pairwise ranking loss across all training instances:

$$\mathcal{L} = \sum_{\langle f,e\rangle} \mathcal{L}(f, e) \tag{6}$$

We used standard back-propagation algorithm to further fine-tune the neural network parameters W and $\mathbf{b}$ in Equation (1). The learned neural networks are used to obtain sentence topic representations, which will be further leveraged to infer topic representations of bilingual translation rules.

3.3 Integration into SMT decoding

We incorporate the learned topic similarity scores into the standard log-linear framework for SMT. When a synchronous rule $\langle \alpha, \gamma\rangle$ is extracted from a sentence pair $\langle f, e\rangle$, a triple instance $\mathcal{I} = (\langle \alpha, \gamma\rangle, \langle f, e\rangle, c)$ is collected for inferring the topic representation of $\langle \alpha, \gamma\rangle$, where c is the count of rule occurrence. Following (Chiang, 2007), we give a count of one for each phrase pair occurrence and a fractional count for each hierarchical phrase pair. The topic representation of $\langle \alpha, \gamma\rangle$ is then calculated as the weighted average:

$$\mathbf{z}_\alpha = \frac{\sum_{(\langle \alpha,\gamma\rangle,\langle f,e\rangle,c)\in\mathcal{T}} \{c \times \mathbf{z}_f\}}{\sum_{(\langle \alpha,\gamma\rangle,\langle f,e\rangle,c)\in\mathcal{T}} \{c\}} \tag{7}$$

$$\mathbf{z}_\gamma = \frac{\sum_{(\langle \alpha,\gamma\rangle,\langle f,e\rangle,c)\in\mathcal{T}} \{c \times \mathbf{z}_e\}}{\sum_{(\langle \alpha,\gamma\rangle,\langle f,e\rangle,c)\in\mathcal{T}} \{c\}} \tag{8}$$

where $\mathcal{T}$ denotes all instances for the rule $\langle \alpha, \gamma\rangle$, $\mathbf{z}_\alpha$ and $\mathbf{z}_\gamma$ are the source-side and target-side topic vectors respectively.

By measuring the similarity between the source texts and bilingual translation rules, the SMT decoder is able to encourage topic relevant translation candidates and penalize topic irrelevant candidates. Therefore, it helps to train a smarter translation model with the embedded topic information. Given a source sentence s to be translated, we define the similarity as follows:

$$Sim(\mathbf{z}_s, \mathbf{z}_\alpha) = \cos(\mathbf{z}_s, \mathbf{z}_\alpha) \tag{9}$$

$$Sim(\mathbf{z}_s, \mathbf{z}_\gamma) = \cos(\mathbf{z}_s, \mathbf{z}_\gamma) \tag{10}$$

where $\mathbf{z}_s$ is the topic representation of s. The similarity calculated against $\mathbf{z}_\alpha$ or $\mathbf{z}_\gamma$ denotes the source-to-source or the source-to-target similarity.

We also consider the topic sensitivity estimation since general rules have flatter distributions while topic-specific rules have sharper distributions. A standard entropy metric is used to measure the sensitivity of the source-side of $\langle \alpha, \gamma\rangle$ as:

$$Sen(\alpha) = -\sum_{i=1}^{|\mathbf{z}_\alpha|} z_{\alpha i} \times \log z_{\alpha i} \tag{11}$$

where $z_{\alpha i}$ is a component in the vector $\mathbf{z}_\alpha$. The target-side sensitivity $Sen(\gamma)$ can be calculated in a similar way. The larger the sensitivity is, the more topic-specific the rule manifests.

In addition to traditional SMT features, we add new topic-related features into the standard log-linear framework. For the SMT system, the best translation candidate $\hat{e}$ is given by:

$$\hat{e} = \arg\max_e P(e|f) \tag{12}$$

where the translation probability is given by:

$$\begin{aligned} P(e|f) &\propto \sum_i w_i \cdot \log \phi_i(f, e) \\ &= \underbrace{\sum_j w_j \cdot \log \phi_j(f, e)}_{\text{Standard}} + \underbrace{\sum_k w_k \cdot \log \phi_k(f, e)}_{\text{Topic related}} \end{aligned} \tag{13}$$

where $\phi_j(f, e)$ is the standard feature function and w_j is the corresponding feature weight. $\phi_k(f, e)$ is the topic-related feature function and w_k is the feature weight. The detailed feature description is as follows:

Standard features: Translation model, including translation probabilities and lexical weights for both directions (4 features), 5-gram language model (1 feature), word count (1 feature), phrase count (1 feature), NULL penalty (1 feature), number of hierarchical rules used (1 feature).

Topic-related features: rule similarity scores (2 features), rule sensitivity scores (2 features).

4 Experiments

4.1 Setup

We evaluate the performance of our neural network based topic similarity model on a Chinese-to-English machine translation task. In neural network training, a large number of monolingual documents are collected in both source and target languages. The documents are mainly from two domains: news and weblog. We use Chinese and

English Gigaword corpus (Version 5) which are mainly from news domain. In addition, we also collect weblog documents with a variety of topics from the web. The total data statistics are presented in Table 1. These documents are built in the format of inverted index using Lucene[2], which can be efficiently retrieved by the parallel sentence pairs. The most relevant N documents are collected, where we experiment with $N = \{1, 5, 10, 20, 50\}$.

Domain	Chinese		English	
	Docs	Words	Docs	Words
News	5.7M	5.4B	9.9M	25.6B
Weblog	2.1M	8B	1.2M	2.9B
Total	7.8M	13.4B	11.1M	28.5B

Table 1: Statistics of monolingual data, in numbers of documents and words (main content). "M" refers to million and "B" refers to billion.

We implement a distributed framework to speed up the training process of neural networks. The network is learned with mini-batch asynchronous gradient descent with the adaptive learning rate procedure called AdaGrad (Duchi et al., 2011). We use 32 model replicas in each iteration during the training. The model parameters are averaged after each iteration and sent to each replica for the next iteration. The vocabulary size for the input layer is 100,000, and we choose different lengths for the hidden layer as $L = \{100, 300, 600, 1000\}$ in the experiments. In the pre-training phase, all parallel data is fed into two neural networks respectively for DAE training, where network parameters W and $\mathbf{b}$ are randomly initialized. In the fine-tuning phase, for each parallel sentence pair, we randomly select other ten sentence pairs which satisfy the criterion as negative instances. These training instances are leveraged to optimize the similarity of two vectors.

In SMT training, an in-house hierarchical phrase-based SMT decoder is implemented for our experiments. The CKY decoding algorithm is used and cube pruning is performed with the same default parameter settings as in Chiang (2007). The parallel data we use is released by LDC[3]. In total, the datasets contain nearly 1.1 million sentence pairs. Translation models are trained over the parallel data that is automatically word-aligned using GIZA++ in both directions, and the diag-grow-final heuristic is used to refine symmetric word alignment. An in-house language modeling toolkit is used to train the 5-gram language model with modified Kneser-Ney smoothing (Kneser and Ney, 1995). The English monolingual data used for language modeling is the same as in Table 1. The NIST 2003 dataset is the development data. The testing data consists of NIST 2004, 2005, 2006 and 2008 datasets. The evaluation metric for the overall translation quality is case-insensitive BLEU4 (Papineni et al., 2002). The reported BLEU scores are averaged over 5 times of running MERT (Och, 2003). A statistical significance test is performed using the bootstrap re-sampling method (Koehn, 2004).

[2]http://lucene.apache.org/

[3]LDC2003E14, LDC2002E18, LDC2003E07, LDC2005T06, LDC2005T10, LDC2005E83, LDC2006E34, LDC2006E85, LDC2006E92, LDC2006E26, LDC2007T09

4.2 Baseline

The baseline is a re-implementation of the Hiero system (Chiang, 2007). The phrase pairs that appear only once in the parallel data are discarded because most of them are noisy. We also use the fix-discount method in Foster et al. (2006) for phrase table smoothing. This implementation makes the system perform much better and the translation model size is much smaller.

We compare our method with the LDA-based approach proposed by Xiao et al. (2012). In (Xiao et al., 2012), the topic of each sentence pair is exactly the same as the document it belongs to. Since some of our parallel data does not have document-level information, we rely on the IR method to retrieve the most relevant document and simulate this approach. The PLDA toolkit (Liu et al., 2011) is used to infer topic distributions, which takes 34.5 hours to finish.

4.3 Effect of retrieved documents and length of hidden layers

We illustrate the relationship among translation accuracy (BLEU), the number of retrieved documents (N) and the length of hidden layers (L) on different testing datasets. The results are shown in Figure 3. The best translation accuracy is achieved when N=10 for most settings. This confirms that enriching the source text with topic-related documents is very useful in determining topic representations, thereby help to guide the synchronous rule selection. However, we find that as N becomes larger in the experiments, e.g. N=50, the translation accuracy drops drastically. As more documents are retrieved, less relevant information

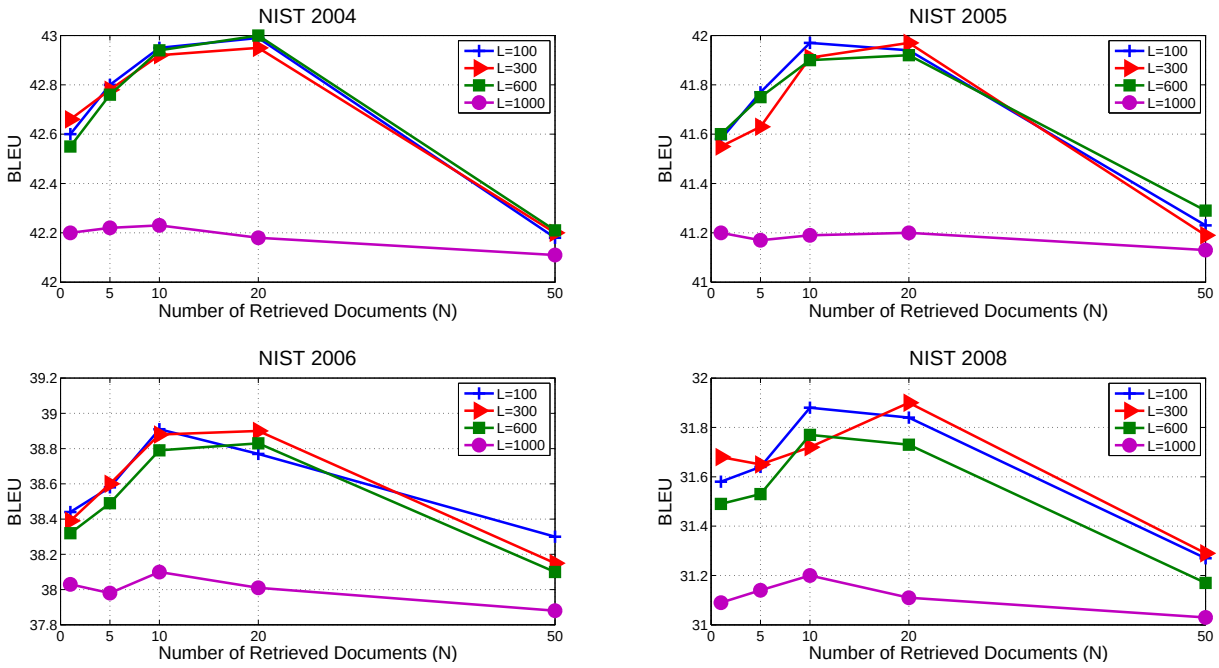


Figure 3: End-to-end translation results (BLEU%) using all standard and topic-related features, with different settings on the number of retrieved documents N and the length of hidden layers L.

is also used to train the neural networks. Irrelevant documents bring so many unrelated topic words hence degrade neural network learning performance.

Another important factor is the length of hidden layers L in the network. In deep learning, this parameter is often empirically tuned with human efforts. As shown in Figure 3, the translation accuracy is better when L is relatively small. Actually, there is no obvious distinction of the performance when L is less than 600. However, when L equals 1,000, the translation accuracy is inferior to other settings. The main reason is that parameters in the neural networks are too many to be effectively trained. As we know when L=1000, there are a total of $100,000 \times 1,000$ parameters between the linear and non-linear layers in the network. Limited training data prevents the model from getting close to the global optimum. Therefore, the model is likely to fall in local optima and lead to unacceptable representations.

4.4 Effect of topic related features

We evaluate the performance of adding new topic-related features to the log-linear model and compare the translation accuracy with the method in (Xiao et al., 2012). To make different methods comparable, we set the dimension of topic representation as 100 for all settings. This takes 10 hours in pre-training phase and 22 hours in fine-tuning phase. Table 2 shows how the accuracy is improved with more features added. The results confirm that topic information is indispensable for SMT since both (Xiao et al., 2012) and our neural network based method significantly outperforms the baseline system. Our method improves 0.86 BLEU points at most and 0.76 BLEU points on average over the baseline. We observe that source-side similarity is more effective than target-side similarity, but their contributions are cumulative. This proves that bilingually induced topic representation with neural network helps the SMT system disambiguate translation candidates. Furthermore, rule sensitivity features improve SMT performance compared with only using similarity features. Because topic-specific rules usually have a larger sensitivity score, they can beat general rules when they obtain the same similarity score against the input sentence. Finally, when all new features are integrated, the performance is the best, preforming substantially better than (Xiao et al., 2012) with 0.39 BLEU points on average.

It is worth mentioning that the performance of (Xiao et al., 2012) is similar to the settings with N=1 and L=100 in Figure 3. This is not simply coincidence since we can interpret their approach as a special case in our neural network method: when a parallel sentence pair has

Settings	NIST 2004	NIST 2005	NIST 2006	NIST 2008	Average
Baseline	42.25	41.21	38.05	31.16	38.17
(Xiao et al., 2012)	42.58	41.61	38.39	31.58	38.54
Sim(Src)	42.51	41.55	38.53	31.57	38.54
Sim(Trg)	42.43	41.48	38.4	31.49	38.45
Sim(Src+Trg)	42.7	41.66	38.66	31.66	38.67
Sim(Src+Trg)+Sen(Src)	42.77	41.81	38.85	31.73	38.79
Sim(Src+Trg)+Sen(Trg)	42.85	41.79	38.76	31.7	38.78
Sim(Src+Trg)+Sen(Src+Trg)	**42.95**	**41.97**	**38.91**	**31.88**	**38.93**

Table 2: Effectiveness of different features in BLEU% ($p < 0.05$), with N=10 and L=100. "Sim" denotes the rule similarity feature and "Sen" denotes rule sensitivity feature. "Src" and "Trg" means utilizing source-side/target-side rule topic vectors to calculate similarity or sensitivity, respectively. The "Average" setting is the averaged result of four datasets.

document-level information, that document will be retrieved for training; otherwise, the most relevant document will be retrieved from the monolingual data. Therefore, our method can be viewed as a more general framework than previous LDA-based approaches.

4.5 Discussion

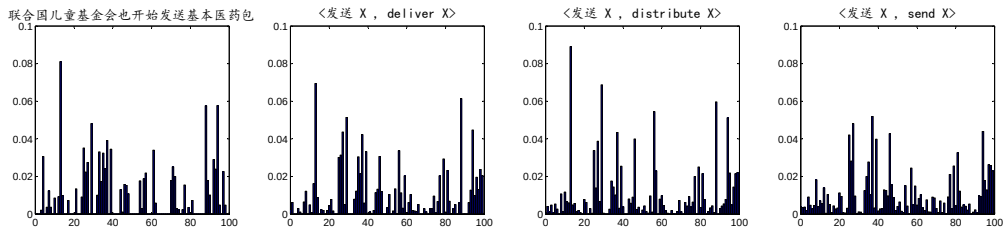
In this section, we give a case study to explain why our method works. An example of translation rule disambiguation for a sentence from the NIST 2005 dataset is shown in Figure 4. We find that the topic of this sentence is about "rescue after a natural disaster". Under this topic, the Chinese rule "发送 X" should be translated to "deliver X" or "distribute X". However, the baseline system prefers "send X" rather than those two candidates. Although the translation probability of "send X" is much higher, it is inappropriate in this context since it is usually used in IT texts. For example, ⟨发送邮件, send emails⟩, ⟨发送信息, send messages⟩ and ⟨发送数据, send data⟩. In contrast, with our neural network based approach, the learned topic distributions of "deliver X" or "distribute X" are more similar with the input sentence than "send X", which is shown in Figure 4. The similarity scores indicate that "deliver X" and "distribute X" are more appropriate to translate the sentence. Therefore, adding topic-related features is able to keep the topic consistency and substantially improve the translation accuracy.

5 Related Work

Topic modeling was first leveraged to improve SMT performance in (Zhao and Xing, 2006; Zhao and Xing, 2007). They proposed a bilingual topical admixture approach for word alignment and assumed that each word-pair follows a topic-specific model. They reported extensive empirical analysis and improved word alignment accuracy as well as translation quality. Following this work, (Xiao et al., 2012) extended topic-specific lexicon translation models to hierarchical phrase-based translation models, where the topic information of synchronous rules was directly inferred with the help of document-level information. Experiments show that their approach not only achieved better translation performance but also provided a faster decoding speed compared with previous lexicon-based LDA methods.

Another direction of approaches leveraged topic modeling techniques for domain adaptation. Tam et al. (2007) used bilingual LSA to learn latent topic distributions across different languages and enforce one-to-one topic correspondence during model training. They incorporated the bilingual topic information into language model adaptation and lexicon translation model adaptation, achieving significant improvements in the large-scale evaluation. (Su et al., 2012) investigated the relationship between out-of-domain bilingual data and in-domain monolingual data via topic mapping using HTMM methods. They estimated phrase-topic distributions in translation model adaptation and generated better translation quality. Recently, Chen et al. (2013) proposed using vector space model for adaptation where genre resemblance is leveraged to improve translation accuracy. We also investigated multi-domain adaptation where explicit topic information is used to train domain specific models (Cui et al., 2013).

Generally, most previous research has leveraged conventional topic modeling techniques such as LDA or HTMM. In our work, a novel neural network based approach is proposed to infer topic representations for parallel data. The advantage of



Rules	$P(\gamma\|\alpha)$	$Sim(\mathbf{z}_s, \mathbf{z}_\alpha)$
⟨发送 X, deliver X⟩	0.0237	0.8469
⟨发送 X, distribute X⟩	0.0546	0.8268
⟨发送 X, send X⟩	0.2464	0.6119

Src	联合国儿童基金会也开始发送基本医药包
Ref	(1) the united nations children's fund has also begun *delivering* basic medical kits (2) the unicef has also started to *distribute* basic medical kits (3) the united nations children's fund has also begun *distributing* basic medical kits (4) the united nations children's fund has begun *delivering* basic medical kits
Baseline	the united nations children's fund began to *send* basic medical kits
Ours	the united nations children's fund has begun to *distribute* basic medical kits

Figure 4: An example from the NIST 2005 dataset. We illustrate the normalized topic representations of the source sentence and three ambiguous synchronous rules. Details are explained in Section 4.5.

our method is that it is applicable to both sentence-level and document-level SMT, since we do not place any restrictions on the input. In addition, our method directly maximizes the similarity between parallel sentence pairs, which is ideal for SMT decoding. Compared to document-level topic modeling which uses the topic of a document for all sentences within the document (Xiao et al., 2012), our contributions are:

- We proposed a more general approach to leveraging topic information for SMT by using IR methods to get a collection of related documents, regardless of whether or not document boundaries are explicitly given.
- We used neural networks to learn topic representations more accurately, with more practicable and scalable modeling techniques.
- We directly optimized bilingual topic similarity in the deep learning framework with the help of sentence-level parallel data, so that the learned representation could be easily used in SMT decoding procedure.

6 Conclusion and Future Work

In this paper, we propose a neural network based approach to learning bilingual topic representation for SMT. We enrich contexts of parallel sentence pairs with topic related monolingual data and obtain a set of documents to represent sentences. These documents are converted to a bag-of-words input and fed into neural networks. The learned low-dimensional vector is used to obtain the topic representations of synchronous rules. In SMT decoding, appropriate rules are selected to best match source texts according to their similarity in the topic space. Experimental results show that our approach is promising for SMT systems to learn a better translation model. It is a significant improvement over the state-of-the-art Hiero system, as well as a conventional LDA-based method.

In the future research, we will extend our neural network methods to address document-level translation, where topic transition between sentences is a crucial problem to be solved. Since the translation of the current sentence is usually influenced by the topic of previous sentences, we plan to leverage recurrent neural networks to model this phenomenon, where the history translation information is naturally combined in the model.

Acknowledgments

We are grateful to the anonymous reviewers for their insightful comments. We also thank Fei Huang (BBN), Nan Yang, Yajuan Duan, Hong Sun and Duyu Tang for the helpful discussions. This work is supported by the National Natural Science Foundation of China (Granted No. 61272384)

References

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. 2006. Greedy layer-wise training of deep networks. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 153–160. MIT Press, Cambridge, MA.

Yoshua Bengio. 2009. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2(1):1–127, January.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March.

Marine Carpuat and Dekai Wu. 2005. Word sense disambiguation vs. statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 387–394, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Marine Carpuat and Dekai Wu. 2007. Context-dependent phrasal translation lexicons for statistical machine translation. *Proceedings of Machine Translation Summit XI*, pages 73–80.

Boxing Chen, Roland Kuhn, and George Foster. 2013. Vector space model for adaptation in statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1285–1293, Sofia, Bulgaria, August. Association for Computational Linguistics.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November.

Lei Cui, Xilun Chen, Dongdong Zhang, Shujie Liu, Mu Li, and Ming Zhou. 2013. Multi-domain adaptation for SMT using multi-task learning. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1055–1065, Seattle, Washington, USA, October. Association for Computational Linguistics.

George E. Dahl, Dong Yu, Li Deng, and Alex Acero. 2012. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech and Language Processing*, 20(1):30–42, January.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July.

George Foster, Roland Kuhn, and Howard Johnson. 2006. Phrasetable smoothing for statistical machine translation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 53–61, Sydney, Australia, July. Association for Computational Linguistics.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics. JMLR W&CP Volume*, volume 15, pages 315–323.

Amit Gruber, Michal Rosen-zvi, and Yair Weiss. 2007. Hidden topic markov models. In *In Proceedings of Artificial Intelligence and Statistics*.

Zhongjun He, Qun Liu, and Shouxun Lin. 2008. Improving statistical machine translation using lexicalized rule selection. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 321–328, Manchester, UK, August. Coling 2008 Organizing Committee.

Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, July.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain, July. Association for Computational Linguistics.

Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. 2012. Imagenet classification with deep convolutional neural networks. In P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1106–1114.

Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y. Ng. 2006. Efficient sparse coding algorithms. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 801–808. MIT Press, Cambridge, MA.

Qun Liu, Zhongjun He, Yang Liu, and Shouxun Lin. 2008. Maximum entropy based rule selection model for syntax-based statistical machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 89–97, Honolulu, Hawaii, October. Association for Computational Linguistics.

Zhiyuan Liu, Yuzhou Zhang, Edward Y. Chang, and Maosong Sun. 2011. Plda+: Parallel latent dirichlet allocation with data placement and pipeline processing. *ACM Transactions on Intelligent Systems and*

Technology, special issue on Large Scale Machine Learning. Software available at `http://code.google.com/p/plda`.

Yuval Marton and Philip Resnik. 2008. Soft syntactic constraints for hierarchical phrased-based translation. In *Proceedings of ACL-08: HLT*, pages 1003–1011, Columbus, Ohio, June. Association for Computational Linguistics.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan, July. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1988. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 354–362, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Richard Socher, Cliff C. Lin, Andrew Y. Ng, and Christopher D. Manning. 2011a. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011b. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 151–161, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Jinsong Su, Hua Wu, Haifeng Wang, Yidong Chen, Xiaodong Shi, Huailin Dong, and Qun Liu. 2012. Translation model adaptation for statistical machine translation with monolingual topic information. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 459–468, Jeju Island, Korea, July. Association for Computational Linguistics.

Yik-Cheung Tam, Ian Lane, and Tanja Schultz. 2007. Bilingual lsa-based adaptation for statistical machine translation. *Machine Translation*, 21(4):187–207, December.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 1096–1103, New York, NY, USA. ACM.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, December.

Xinyan Xiao, Deyi Xiong, Min Zhang, Qun Liu, and Shouxun Lin. 2012. A topic similarity model for hierarchical phrase-based translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 750–758, Jeju Island, Korea, July. Association for Computational Linguistics.

Deyi Xiong and Min Zhang. 2013. A topic-based coherence model for statistical machine translation. In *AAAI*.

Deyi Xiong, Min Zhang, Aiti Aw, and Haizhou Li. 2009. A syntax-driven bracketing model for phrase-based translation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 315–323, Suntec, Singapore, August. Association for Computational Linguistics.

Bing Zhao and Eric P. Xing. 2006. Bitam: Bilingual topic admixture models for word alignment. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 969–976, Sydney, Australia, July. Association for Computational Linguistics.

Bing Zhao and Eric P. Xing. 2007. Hm-bitam: Bilingual topic exploration, word alignment, and translation. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1689–1696. MIT Press, Cambridge, MA.

Tagging The Web: Building A Robust Web Tagger with Neural Network

Ji Ma[†], **Yue Zhang**[‡] **and Jingbo Zhu**[†]
[†]Northeastern University, China
[‡]Singapore University of Technology and Design
majineu@gmail.com
yue_zhang@sutd.edu.sg
zhujingbo@mail.neu.edu.cn

Abstract

In this paper, we address the problem of web-domain POS tagging using a two-phase approach. The first phase learns representations that capture regularities underlying web text. The representation is integrated as features into a neural network that serves as a scorer for an easy-first POS tagger. Parameters of the neural network are trained using guided learning in the second phase. Experiment on the SANCL 2012 shared task show that our approach achieves 93.15% average tagging accuracy, which is the best accuracy reported so far on this data set, higher than those given by ensembled syntactic parsers.

1 Introduction

Analysing and extracting useful information from the web has become an increasingly important research direction for the NLP community, where many tasks require part-of-speech (POS) tagging as a fundamental preprocessing step. However, state-of-the-art POS taggers in the literature (Collins, 2002; Shen et al., 2007) are mainly optimized on the the Penn Treebank (PTB), and when shifted to web data, tagging accuracies drop significantly (Petrov and McDonald, 2012).

The problem we face here can be considered as a special case of *domain adaptation*, where we have access to labelled data on the source domain (PTB) and unlabelled data on the target domain (web data). Exploiting useful information from the web data can be the key to improving web domain tagging. Towards this end, we adopt the idea of learning representations which has been demonstrated useful in capturing hidden regularities underlying the raw input data (web text, in our case).

Our approach consists of two phrases. In the pre-training phase, we learn an encoder that converts the web text into an intermediate representation, which acts as useful features for prediction tasks. We integrate the learned encoder with a set of well-established features for POS tagging (Ratnaparkhi, 1996; Collins, 2002) in a single neural network, which is applied as a scorer to an easy-first POS tagger. We choose the easy-first tagging approach since it has been demonstrated to give higher accuracies than the standard left-to-right POS tagger (Shen et al., 2007; Ma et al., 2013).

In the fine-tuning phase, the parameters of the network are optimized on a set of labelled training data using guided learning. The learned model preserves the property of preferring to tag *easy* words first. To our knowledge, we are the first to investigate guided learning for neural networks.

The idea of learning representations from unlabelled data and then fine-tuning a model with such representations according to some supervised criterion has been studied before (Turian et al., 2010; Collobert et al., 2011; Glorot et al., 2011). While most previous work focus on in-domain sequential labelling or cross-domain classification tasks, we are the first to learn representations for web-domain structured prediction. Previous work treats the learned representations either as model parameters that are further optimized in supervised fine-tuning (Collobert et al., 2011) or as fixed features that are kept unchanged (Turian et al., 2010; Glorot et al., 2011). In this work, we investigate both strategies and give empirical comparisons in the cross-domain setting. Our results suggest that while both strategies improve in-domain tagging accuracies, keeping the learned representation unchanged consistently results in better cross-domain accuracies.

We conduct experiments on the official data set provided by the SANCL 2012 shared task (Petrov and McDonald, 2012). Our method achieves a 93.15% average accuracy across the web-domain, which is the best result reported so far on this data

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 144–154,
Baltimore, Maryland, USA, June 23-25 2014.

set, higher than those given by ensembled syntactic parsers. Our code will be publicly available at https://github.com/majineu/TWeb.

2 Learning from Web Text

Unsupervised learning is often used for training encoders that convert the input data to abstract representations (i.e. encoding vectors). Such representations capture hidden properties of the input, and can be used as features for supervised tasks (Bengio, 2009; Ranzato et al., 2007). Among the many proposed encoders, we choose the restricted Boltzmann machine (RBM), which has been successfully used in many tasks (Lee et al., 2009b; Hinton et al., 2006). In this section, we give some background on RBMs and then show how they can be used to learn representations of the web text.

2.1 Restricted Boltzmann Machine

The RBM is a type of graphical model that contains two layers of binary stochastic units $\mathbf{v} \in \{0,1\}^V$ and $\mathbf{h} \in \{0,1\}^H$, corresponding to a set of visible and hidden variables, respectively. The RBM defines the joint probability distribution over $\mathbf{v}$ and $\mathbf{h}$ by an energy function

$$E(\mathbf{v},\mathbf{h}) = -\mathbf{c}'\mathbf{h} - \mathbf{b}'\mathbf{v} - \mathbf{h}'\mathbf{W}\mathbf{v}, \qquad (1)$$

which is factorized by a visible bias $\mathbf{b} \in \mathbb{R}^V$, a hidden bias $\mathbf{c} \in \mathbb{R}^H$ and a weight matrix $\mathbf{W} \in \mathbb{R}^{H\times V}$. The joint distribution $P(\mathbf{v},\mathbf{h})$ is given by

$$P(\mathbf{v},\mathbf{h}) = \frac{1}{Z}\exp(E(\mathbf{v},\mathbf{h})), \qquad (2)$$

where Z is the partition function.

The affine form of E with respect to $\mathbf{v}$ and $\mathbf{h}$ implies that the visible variables are conditionally independent with each other given the hidden layer units, and vice versa. This yields the conditional distribution:

$$P(\mathbf{v}|\mathbf{h}) = \prod_{j=1}^{V} P(v_j|\mathbf{h}) \quad P(\mathbf{h}|\mathbf{v}) = \prod_{i=1}^{H} P(h_i|\mathbf{v})$$

$$P(v_j = 1|\mathbf{h}) = \sigma(\mathbf{b}_j + W_{.j}\mathbf{h}) \qquad (3)$$

$$P(h_i = 1|\mathbf{v}) = \sigma(\mathbf{c}_j + W_{i.}\mathbf{v}) \qquad (4)$$

Here σ denotes the sigmoid function. Parameters of RBMs $\theta = \{\mathbf{b},\mathbf{c},\mathbf{W}\}$ can be trained efficiently using contrastive divergence learning (CD), see (Hinton, 2002) for detailed descriptions of CD.

2.2 Encoding Web Text with RBM

Most of the indicative features for POS disambiguation can be found from the words and word combinations within a local context (Ratnaparkhi, 1996; Collins, 2002). Inspired by this observation, we apply the RBM to learn feature representations from word n-grams. More specifically, given the i^{th} word w_i of a sentence, we apply RBMs to model the joint distribution of the n-gram $(w_{i-l},\cdots,w_{i+r})$, where l and r denote the left and right window, respectively. Note that the visible units of RBMs are binary. While in our case, each visible variable corresponds to a word, which may take on tens-of-thousands of different values. Therefore, the RBM need to be re-factorized to make inference tractable.

We utilize the Word Representation RBM (WRRBM) factorization proposed by Dahl et al. (2012). The basic idea is to share word representations across different positions in the input n-gram while using position-dependent weights to distinguish between different word orders.

Let w_k be the k-th entry of lexicon L, and $\mathbf{w}_k$ be its *one-hot* representation (i.e., only the k-th component of $\mathbf{w}_k$ is 1, and all the others are 0). Let $\mathbf{v}^{(j)}$ represents the j-th visible variable of the WRRBM, which is a vector of length $|L|$. Then $\mathbf{v}^{(j)} = \mathbf{w}_k$ means that the j-th word in the n-gram is w_k. Let $\mathbf{D} \in \mathbb{R}^{D\times|L|}$ be a projection matrix, then $\mathbf{D}\mathbf{w}_k$ projects w_k into a D-dimensional real value vector (embedding). For each position j, there is a weight matrix $\mathbf{W}^{(j)} \in \mathbb{R}^{H\times D}$, which is used to model the interaction between the hidden layer and the word projection in position j. The visible biases are also shared across different positions ($b^{(j)} = b\ \forall j$) and the energy function is:

$$E(\mathbf{v},\mathbf{h}) = -\mathbf{c}'\mathbf{h} - \sum_{j=1}^{n}(\mathbf{b}'\mathbf{v}^{(j)} + \mathbf{h}'\mathbf{W}^{(j)}\mathbf{D}\mathbf{v}^{(j)}), \qquad (5)$$

which yields the conditional distributions:

$$P(\mathbf{v}|\mathbf{h}) = \prod_{j=1}^{n} P(\mathbf{v}^{(j)}|\mathbf{h}) \quad P(\mathbf{h}|\mathbf{v}) = \prod_{i=1} P(h_i|\mathbf{v})$$

$$P(h_i = 1|\mathbf{v}) = \sigma(c_i + \sum_{j=1}^{n}\mathbf{W}_{i.}^{(j)}\mathbf{D}\mathbf{v}^{(j)}) \qquad (6)$$

$$P(\mathbf{v}^{(j)} = \mathbf{w}_k|\mathbf{h}) = \frac{1}{Z}\exp(\mathbf{b}'\mathbf{w}_k + \mathbf{h}'\mathbf{W}^{(j)}\mathbf{D}\mathbf{w}_k) \qquad (7)$$

Again Z is the partition function.

The parameters $\{\mathbf{b}, \mathbf{c}, \mathbf{D}, \mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(n)}\}$ can be trained using a Metropolis-Hastings-based CD variant and the learned word representations also capture certain syntactic information; see Dahl et al. (2012) for more details.

Note that one can stack standard RBMs on top of a WRRBM to construct a Deep Belief Network (DBN). By adopting greedy layer-wise training (Hinton et al., 2006; Bengio et al., 2007), DBNs are capable of modelling higher order non-linear relations between the input, and has been demonstrated to improve performance for many computer vision tasks (Hinton et al., 2006; Bengio et al., 2007; Lee et al., 2009a). However, in this work we do not observe further improvement by employing DBNs. This may partly be due to the fact that unlike computer vision tasks, the input structure of POS tagging or other sequential labelling tasks is relatively simple, and a single non-linear layer is enough to model the interactions within the input (Wang and Manning, 2013).

3 Neural Network for POS Disambiguation

We integrate the learned WRRBM into a neural network, which serves as a scorer for POS disambiguation. The main challenge to designing the neural network structure is: on the one hand, we hope that the model can take the advantage of information provided by the learned WRRBM, which reflects general properties of web texts, so that the model generalizes well in the web domain; on the other hand, we also hope to improve the model's discriminative power by utilizing well-established POS tagging features, such as those of Ratnaparkhi (1996).

Our approach is to leverage the two sources of information in one neural network by combining them though a shared output layer, as shown in Figure 1. Under the output layer, the network consists of two modules: *the web-feature module*, which incorporates knowledge from the pre-trained WRRBM, and *the sparse-feature module*, which makes use of other POS tagging features.

3.1 The Web-Feature Module

The web-feature module, shown in the lower left part of Figure 1, consists of a input layer and two hidden layers. The input for the this module is the word n-gram $(w_{i-l}, \ldots, w_{i+r})$, the form of which

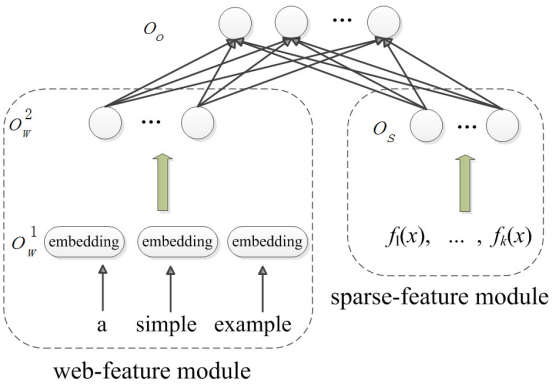


Figure 1: The proposed neural network. The web-feature module (lower left) and sparse-feature module (lower right) are combined by a shared output layer (upper).

is identical to the training data of the pre-trained WRRBM.

The first layer is a linear projection layer, where each word in the input is projected into a D-dimensional real value vector using the projection operation described in Section 2.2. The output of this layer $\mathbf{o}_w^1$ is the concatenation of the projections of $w_{i-l}, \ldots, w_{i+r}$:

$$\mathbf{o}_w^1 = \begin{pmatrix} \mathbf{M}_w^1 \mathbf{w}_{i-l} \\ \vdots \\ \mathbf{M}_w^1 \mathbf{w}_{i+r} \end{pmatrix} \tag{8}$$

Here $\mathbf{M}_w^1$ denotes the parameters of the first layer of the web-feature module, which is a $D \times |L|$ projection matrix.

The second layer is a sigmoid layer to model non-linear relations between the word projections:

$$\mathbf{o}_w^2 = \sigma(\mathbf{M}_w^2 \mathbf{o}_w^1 + \mathbf{b}_w^2) \tag{9}$$

Parameters of this layer include: a bias vector $\mathbf{b}_w^2 \in \mathbb{R}^H$ and a weight matrix $\mathbf{M}_w^2 \in \mathbb{R}^{H \times nD}$.

The web-feature module enables us to explore the learned WRRBM in various ways. First, it allows us to investigate knowledge from the WRRBM incrementally. We can choose to use only the *word* representations of the learned WRRBM. This can be achieved by initializing only the first layer of the web module with the projection matrix $\mathbf{D}$ of the learned WRRBM:

$$\mathbf{M}_w^1 \leftarrow \mathbf{D}. \tag{10}$$

Alternatively, we can choose to use the hidden states of the WRRBM, which can be treated as the

representations of the input *n-gram*. This can be achieved by *also* initializing the parameters of the second layer of the web-feature module using the position-dependent weight matrix and hidden bias of the learned WRRBM:

$$\mathbf{b}_w^2 \leftarrow \mathbf{c} \tag{11}$$

$$\mathbf{M}_w^2 \leftarrow (\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(n)}) \tag{12}$$

Second, the web-feature module also allows us to make a comparison between whether or not to further adjust the pre-trained representation in the supervised fine-tuning phase, which corresponds to the supervised learning strategies of Turian et al. (2010) and Collobert et al. (2011), respectively. To our knowledge, no investigations have been presented in the literature on this issue.

3.2 The Sparse-Feature Module

The sparse-feature module, as shown in the lower right part of Figure 1, is designed to incorporate commonly-used tagging features. The input for this module is a vector of boolean values $\Phi(x) = (f_1(x), \ldots, f_k(x))$, where x denotes the partially tagged input sentence and $f_i(x)$ denotes a feature function, which returns 1 if the corresponding feature fires and 0 otherwise. The first layer of this module is a linear transformation layer, which converts the high dimensional sparse vector into a fixed-dimensional real value vector:

$$\mathbf{o}_s = \mathbf{M}_s \Phi(x) + \mathbf{b}_s \tag{13}$$

Depending on the specific task being considered, the output of this layer can be further fed to other non-linear layers, such as a sigmoid or hyperbolic tangent layer, to model more complex relations. For POS tagging, we found that a simple linear layer yields satisfactory accuracies.

The web-feature and sparse-feature modules are combined by a linear output layer, as shown in the upper part of Figure 1. The value of each unit in this layer denotes the score of the corresponding POS tag.

$$\mathbf{o}_o = \mathbf{M}_o \begin{pmatrix} \mathbf{o}_w \\ \mathbf{o}_s \end{pmatrix} + \mathbf{b}_o \tag{14}$$

In some circumstances, probability distribution over POS tags might be a more preferable form of output. Such distribution can be easily obtained by adding a soft-max layer on top of the output layer to perform a local normalization, as done by Collobert et al. (2011).

Algorithm 1 Easy-first POS tagging

Input: x a sentence of m words $w_1, \ldots, w_m$
Output: tag sequence of x

1: $\mathbf{U} \leftarrow [w_1, \ldots, w_m]$ // untagged words
2: **while** $\mathbf{U} \neq []$ **do**
3: $(\hat{w}, \hat{t}) \leftarrow \arg\max_{(w,t) \in \mathbf{U} \times \mathbf{T}} S(w, t)$
4: $\hat{w}.t \leftarrow \hat{t}$
5: $\mathbf{U} \leftarrow \mathbf{U}/[\hat{w}]$ // remove $\hat{w}$ from $\mathbf{U}$
6: **end while**
7: **return** $[w_1.t, \ldots, w_m.t]$

4 Easy-first POS tagging with Neural Network

The neural network proposed in Section 3 is used for POS disambiguation by the easy-first POS tagger. Parameters of the network are trained using guided learning, where learning and search interact with each other.

4.1 Easy-first POS tagging

Pseudo-code of easy-first tagging is shown in Algorithm 1. Rather than tagging a sentence from left to right, easy-first tagging is based on a deterministic process, repeatedly selecting the *easiest* word to tag. Here "easiness" is evaluated based on a statistical model. At each step, the algorithm adopts a scorer, the neural network in our case, to assign a score to each possible word-tag pair (w, t), and then selects the highest score one $(\hat{w}, \hat{t})$ to tag (i.e., tag $\hat{w}$ with $\hat{t}$). The algorithm repeats until all words are tagged.

4.2 Training

The training algorithm repeats for several iterations over the training data, which is a set of sentences labelled with gold standard POS tags. In each iteration, the procedure shown in Algorithm 2 is applied to each sentence in the training set.

At each step during the processing of a training example, the algorithm calculates a *margin loss* based on two word-tag pairs $(\overline{w}, \overline{t})$ and $(\hat{w}, \hat{t})$ (line 4 $\sim$ line 6). $(\overline{w}, \overline{t})$ denotes the word-tag pair that has the highest model score among those that are *inconsistent* with the gold standard, while $(\hat{w}, \hat{t})$ denotes the one that has the highest model score among those that are *consistent* with the gold standard. If the loss is zero, the algorithm continues to process the next untagged word. Otherwise, parameters are updated using back-propagation.

The standard back-propagation algorithm

(Rumelhart et al., 1988) cannot be applied directly. This is because the standard loss is calculated based on a *unique* input vector. This condition does not hold in our case, because $\hat{w}$ and $\overline{w}$ may refer to different words, which means that the margin loss in line 6 of Algorithm 2 is calculated based on two different input vectors, denoted by $\langle\hat{w}\rangle$ and $\langle\overline{w}\rangle$, respectively.

We solve this problem by decomposing the margin loss in line 6 into two parts:

- $1 + nn(\overline{w}, \bar{t})$, which is associated with $\langle\overline{w}\rangle$;
- $-nn(\hat{w}, \hat{t})$, which is associated with $\langle\hat{w}\rangle$.

In this way, two separate back-propagation updates can be used to update the model's parameters (line 8 $\sim$ line 11). For the special case where $\hat{w}$ and $\overline{w}$ do refer to the same word w, it can be easily verified that the two separate back-propagation updates equal to the standard back-propagation with a loss $1 + nn(w, \bar{t}) - nn(w, \hat{t})$ on the input $\langle w\rangle$.

The algorithm proposed here belongs to a general framework named *guided learning*, where search and learning interact with each other. The algorithm learns not only a local classifier, but also the inference order. While previous work (Shen et al., 2007; Zhang and Clark, 2011; Goldberg and Elhadad, 2010) apply guided learning to train a linear classifier by using variants of the perceptron algorithm, we are the first to combine guided learning with a neural network, by using a margin loss and a modified back-propagation algorithm.

5 Experiments

5.1 Setup

Our experiments are conducted on the data set provided by the SANCL 2012 shared task, which aims at building a single robust syntactic analysis system across the web-domain. The data set consists of labelled data for both the source (Wall Street Journal portion of the Penn Treebank) and target (web) domains. The web domain data can be further classified into five sub-domains, including emails, weblogs, business reviews, news groups and Yahoo!Answers. While emails and weblogs are used as the development sets, reviews, news groups and Yahoo!Answers are used as the final test sets. Participants are not allowed to use web-domain labelled data for training. In addition to labelled data, a large amount of unlabelled data on the web domain is also provided. Statistics

Algorithm 2 Training over one sentence

Input: (x, t) a tagged sentence, neural net nn
Output: updated neural net nn'

1: $\mathbf{U} \leftarrow [w_1, \ldots, w_m]$ // untagged words
2: $\mathbf{R} \leftarrow [(w_1, t_1), \ldots, (w_m, t_m)]$ // reference
3: **while** $\mathbf{U} \neq []$ **do**
4: $\quad(\overline{w}, \bar{t}) \leftarrow \arg\max_{(w,t)\in(\mathbf{U}\times\mathbf{T}/\mathbf{R})} nn(w, t)$
5: $\quad(\hat{w}, \hat{t}) \leftarrow \arg\max_{(w,t)\in\mathbf{R}} nn(w, t)$
6: $\quad loss \leftarrow \max(0, 1 + nn(\overline{w}, \bar{t}) - nn(\hat{w}, \hat{t}))$
7: $\quad$**if** $loss > 0$ **then**
8: $\quad\quad\hat{e} \leftarrow nn$.BackPropErr$(\langle\hat{w}\rangle, -nn(\hat{w}, \hat{t}))$
9: $\quad\quad\overline{e} \leftarrow nn$.BackPropErr$(\langle\overline{w}\rangle, 1+nn(\overline{w}, \bar{t}))$
10: $\quad\quad nn$.Update$(\langle\hat{w}\rangle, \hat{e})$
11: $\quad\quad nn$.Update$(\langle\overline{w}\rangle, \overline{e})$
12: $\quad$**else**
13: $\quad\quad\mathbf{U} \leftarrow \mathbf{U}/\{\hat{w}\}, \quad \mathbf{R} \leftarrow \mathbf{R}/(\hat{w}, \hat{t})$
14: $\quad$**end if**
15: **end while**
16: **return** nn

about labelled and unlabelled data are summarized in Table 1 and Table 2, respectively.

The raw web domain data contains much noise, including spelling error, emotions and inconsistent capitalization. Following some participants (Le Roux et al., 2012), we conduct simple preprocessing steps to the input of the development and the test sets[1]

- Neutral quotes are transformed to opening or closing quotes.
- Tokens starting with "www.", "http." or ending with ".org", ".com" are converted to a "#URL" symbol
- Repeated punctuations such as "!!!!" are collapsed into one.
- Left brackets such as "<","{" and "[" are converted to "-LRB-". Similarly, right brackets are converted to "-RRB-"
- Upper cased words that contain more than 4 letters are lowercased.
- Consecutive occurrences of one or more digits within a word are replaced with "#DIG"

We apply the same preprocessing steps to all the unlabelled data. In addition, following Dahl et

[1]The preprocessing steps make use of no POS knowledge, and does not bring any unfair advantages to the participants.

	Training set	Dev set			Test set			
	WSJ-Train	Emails	Weblogs	WSJ-dev	Answers	Newsgroups	Reviews	WSJ-test
#Sen	30060	2,450	1,016	1,336	1,744	1,195	1,906	1,640
#Words	731,678	29,131	24,025	32,092	28,823	20,651	28,086	35,590
#Types	35,933	5,478	4,747	5,889	4,370	4,924	4,797	6,685

Table 1: Statistics of the labelled data. #Sen denotes number of sentences. #Words and #Types denote number of words and unique word types, respectively.

	Emails	Weblogs	Answers	Newsgroups	Reviews
#Sen	1,194,173	524,834	27,274	1,000,000	1,965,350
#Words	17,047,731	10,365,284	424,299	18,424,657	29,289,169
#Types	221,576	166,515	33,325	357,090	287,575

Table 2: Statistics of the raw unlabelled data.

features	templates
unigram	$H(w_i)$, $C(w_i)$, $L(w_i)$, $L(w_{i-1})$, $L(w_{i+1})$, t_{i-2}, t_{i-1}, t_{i+1}, t_{i+2}
bigram	$L(w_i) \odot L(w_{i-1})$, $L(w_i) \odot L(w_{i+1})$, $t_{i-2} \odot t_{i-1}$, $t_{i-1} \odot t_{i+1}$, $t_{i+1} \odot t_{i+2}$, $L(w_i) \odot t_{i-2}$, $L(w_i) \odot t_{i-1}$, $L(w_i) \odot t_{i+1}$, $L(w_i) \odot t_{i+2}$
trigram	$L(w_i) \odot t_{i-2} \odot t_{i-1}$, $L(w_i) \odot t_{i-1} \odot t_{i+1}$, $L(w_i) \odot t_{i+1} \odot t_{i+2}$

Table 3: Feature templates, where w_i denotes the current word. $H(w)$ and $C(w)$ indicates whether w contains hyphen and upper case letters, respectively. $L(w)$ denotes a lowercased w.

al. (2012) and Turian et al. (2010), we also lowercased all the unlabelled data and removed those sentences that contain less than 90% a-z letters.

The tagging performance is evaluated according to the official evaluation metrics of SANCL 2012. The tagging accuracy is defined as the percentage of words (punctuations included) that are correctly tagged. The averaged accuracies are calculated across the web domain data.

We trained the WRRBM on web-domain data of different sizes (number of sentences). The data sets are generated by first concatenating all the cleaned unlabelled data, then selecting sentences evenly across the concatenated file.

For each data set, we investigate an extensive set of combinations of hyper-parameters: the n-gram window (l, r) in $\{(1,1),(2,1),(1,2),(2,2)\}$; the hidden layer size in $\{200, 300, 400\}$; the learning rate in $\{0.1, 0.01, 0.001\}$. All these parameters are selected according to the averaged accuracy on the development set.

5.2 Baseline

We reimplemented the greedy easy-first POS tagger of Ma et al. (2013), which is used for all the experiments. While the tagger of Ma et al. (2013) utilizes a linear scorer, our tagger adopts the neural network as its scorer. The neural network of our baseline tagger only contains the sparse-feature module. We use this baseline to examine the performance of a tagger trained purely on the source domain. Feature templates are shown in Table 3, which are based on those of Ratnaparkhi (1996) and Shen et al. (2007).

Accuracies of the baseline tagger are shown in the upper part of Table 6. Compared with the performance of the official baseline (row 4 of Table 6), which is evaluated based on the output of BerkeleyParser (Petrov et al., 2006; Petrov and Klein, 2007), our baseline tagger achieves comparable accuracies on both the source and target domain data. With data preprocessing, the average accuracy boosts to about 92.02 on the test set of the target domain. This is consistent with previous work (Le Roux et al., 2011), which found that for noisy data such as web domain text, data cleaning is a effective and necessary step.

5.3 Exploring the Learned Knowledge

As mentioned in Section 3.1, the knowledge learned from the WRRBM can be investigated incrementally, using *word representation*, which corresponds to initializing only the projection layer of web-feature module with the projection matrix of the learned WRRBM, or *ngram-level representation*, which corresponds to initializing both the projection and sigmoid layers of the web-feature module by the learned WRRBM. In each case, there can be two different **training strategies** depending on whether the learned representations are further adjusted or kept unchanged during the fine-turning phrase. Experimental results under the 4 combined settings on the development sets are illustrated in Figure 2, 3 and 4, where the

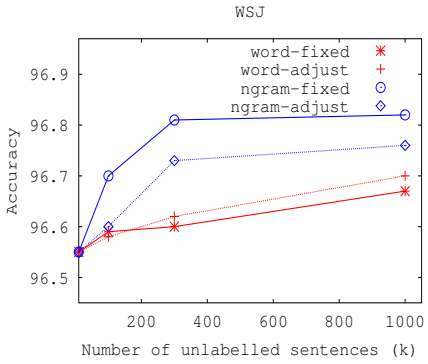


Figure 2: Tagging accuracies on the source-domain data. "word" and "ngram" denote using word representations and n-gram representations, respectively. "fixed" and "adjust" denote that the learned representation are kept unchanged or further adjusted in supervised learning, respectively.

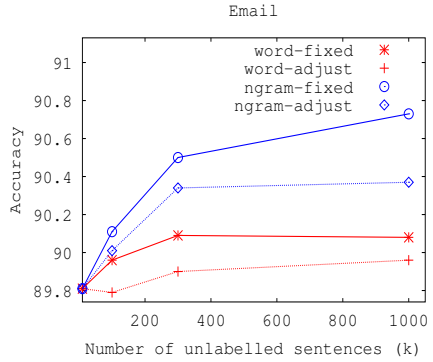


Figure 3: Accuracies on the email domain.

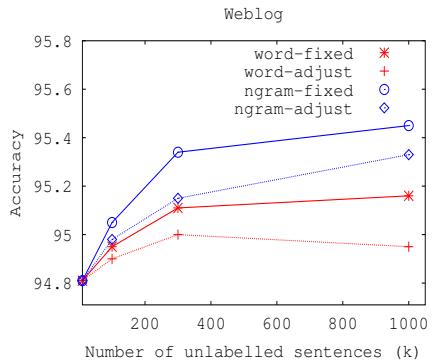


Figure 4: Accuracies on the weblog domain.

x-axis denotes the size of the training data and y-axis denotes tagging accuracy.

5.3.1 Effect of the Training Strategy

From Figure 2 we can see that when knowledge from the pre-trained WRRBM is incorporated, both the **training strategies** ("word-fixed" vs "word-adjusted", "ngram-fixed" vs "ngram-adjusted") improve accuracies on the source domain, which is consistent with previous findings (Turian et al., 2010; Collobert et al., 2011). In addition, adjusting the learned representation or keeping them fixed does not result in too much difference in tagging accuracies.

method	all	non-oov	oov
baseline	89.81	92.42	65.64
word-adjust	+0.09	−0.05	+1.38
word-fix	+0.11	+0.13	+1.73
ngram-adjust	+0.53	+0.52	+0.53
ngram-fix	+0.69	+0.60	+2.30

Table 4: Performance on the email domain.

On the web-domain data, shown in Figure 3 and 4, we found that leaving the learned representation unchanged ("word-fixed", "ngram-fixed") yields consistently higher performance gains. This result is to some degree expected. Intuitively, unsupervised pre-training moves the parameters of the WRRBM towards the region where properties of the *web domain* data are properly modelled. However, since fine-tuning is conducted with respect to the *source domain*, adjusting the parameters of the pre-trained representation towards optimizing source domain tagging accuracies would disrupt its ability in modelling the web domain data. Therefore, a better idea is to keep the representation unchanged so that we can learn a function that maps the general web-text properties to its syntactic categories.

5.3.2 Word and N-gram Representation

From Figures 2, 3 and 4, we can see that adopting the ngram-level representation consistently achieves better performance compared with using word representations only ("word-fixed" vs "ngram-fixed", "word-adjusted" vs "ngram-adjusted"). This result illustrates that the ngram-level knowledge captures more complex interactions of the web text, which cannot be recovered by using only word embeddings. Similar result was reported by Dahl et al. (2012), who found that using both the word embeddings and the hidden units of a tri-gram WRRBM as additional features for a CRF chunker yields larger improvements than using word embeddings only.

Finally, more detailed accuracies under the 4 settings on the email domain are shown in Table 4. We can see that the improvement of using word

		RBM-E	RBM-W	RBM-M
+acc%	Emails	+0.73	+0.37	+0.69
	Weblog	+0.31	+0.52	+0.54
cov%	Emails	95.24	92.79	93.88
	Weblog	90.21	97.74	94.77

Table 5: Effect of unlabelled data. "+acc" denotes improvement in tagging accuracy and "cov" denotes the lexicon coverages.

representations mainly comes from better accuracy of out-of-vocabulary (oov) words. By contrast, using n-gram representations improves the performance on both oov and non-oov.

5.4 Effect of Unlabelled Domain Data

In some circumstances, we may know beforehand that the target domain data belongs to a certain sub-domain, such as the email domain. In such cases, it might be desirable to train WRRBM using data only on that domain. We conduct experiments to test whether using the target domain data to train the WRRBM yields better performance compared with using mixed data from all sub-domains.

We trained 3 WRRBMs using the email domain data (RBM-E), weblog domain data (RBM-W) and mixed domain data (RBM-M), respectively, with each data set consisting of 300k sentences. Tagging performance and lexicon coverages of each data set on the development sets are shown in Table 5. We can see that using the target domain data achieves similar improvements compared with using the mixed data. However, for the email domain, RBM-W yields much smaller improvement compared with RBM-E, and vice versa. From the lexicon coverages, we can see that the sub-domains varies significantly. The results suggest that using mixed data can achieve almost as good performance as using the target sub-domain data, while using mixed data yields a much more robust tagger across all sub-domains.

5.5 Final Results

The best result achieved by using a 4-gram WRRBM, $(w_{i-2}, \ldots, w_{i+1})$, with 300 hidden units learned on 1,000k web domain sentences are shown in row 3 of Table 6. Performance of the top 2 systems of the SANCL 2012 task are also shown in Table 6. Our greedy tagger achieves 93% tagging accuracy, which is significantly better than the baseline's 92.02% accuracy ($p < 0.05$ by McNemar's test). Moreover, we achieve the highest tagging accuracy reported so far on this data set, surpassing those achieved using parser combinations based on self-training (Tang et al., 2012; Le Roux et al., 2012). In addition, different from Le Roux et al. (2012), we do not use any external resources in data cleaning.

6 Related Work

Learning representations has been intensively studied in computer vision tasks (Bengio et al., 2007; Lee et al., 2009a). In NLP, there is also much work along this line. In particular, Collobert et al. (2011) and Turian et al. (2010) learn word embeddings to improve the performance of in-domain POS tagging, named entity recognition, chunking and semantic role labelling. Yang et al. (2013) induce bi-lingual word embeddings for word alignment. Zheng et al. (2013) investigate Chinese character embeddings for joint word segmentation and POS tagging. While those approaches mainly explore token-level representations (word or character embeddings), using WRRBM is able to utilize both word and n-gram representations.

Titov (2011) and Glorot et al. (2011) propose to learn representations from the mixture of both source and target domain unlabelled data to improve cross-domain sentiment classification. Titov (2011) also propose a regularizer to constrain the inter-domain variability. In particular, their regularizer aims to minimize the Kullback-Leibler (KL) distance between the marginal distributions of the learned representations on the source and target domains.

Their work differs from ours in that their approaches learn representations from the feature vectors for sentiment classification, which might be of thousands of dimensions. Such high dimensional input gives rise to high computational cost and it is not clear whether those approaches can be applied to large scale unlabelled data, with hundreds of millions of training examples. Our method learns representations from only word n-grams with n ranging from 3 to 5, which can be easily applied to large scale-data. In addition, while Titov (2011) and Glorot et al. (2011) use the learned representation to improve cross-domain classification tasks, we are the first to apply it to cross-domain structured prediction.

Blitzer et al. (2006) propose to induce shared representations for domain adaptation, which is based on the alternating structure optimization

System	Answer	Newsgroup	Review	WSJ-t	Avg
baseline-raw	89.79	91.36	89.96	97.09	90.31
baseline-clean	91.35	92.06	92.92	97.09	92.02
best-clean	**92.37**	93.59	**93.62**	97.44	**93.15**
baseline-offical	90.20	91.24	89.33	97.08	90.26
Le Roux et al.(2011)	91.79	**93.81**	93.11	97.29	92.90
Tang et al. (2012)	91.76	92.91	91.94	**97.49**	92.20

Table 6: Main results. "baseline-raw" and "baseline-clean" denote performance of our baseline tagger on the raw and cleaned data, respectively. "best-clean" is best performance achieved using a 4-gram WRRBM. The lower part shows accuracies of the official baseline and that of the top 2 participants.

(ASO) method of Ando and Zhang (2005). The idea is to project the original feature representations into low dimensional representations, which yields a high-accuracy classifier on the target domain. The new representations are induced based on the auxiliary tasks defined on unlabelled data together with a dimensionality reduction technique. Such auxiliary tasks can be specific to the supervised task. As pointed out by Plank (2009), for many NLP tasks, defining the auxiliary tasks is a non-trivial engineering problem. Compared with Blitzer et al. (2006), the advantage of using RBMs is that it learns representations in a pure unsupervised manner, which is much simpler.

Besides learning representations, another line of research addresses domain-adaptation by instance re-weighting (Bickel et al., 2007; Jiang and Zhai, 2007) or feature re-weighting (Satpal and Sarawagi, 2007). Those methods assume that each example x that has a non-zero probability on the source domain must have a non-zero probability on the target domain, and vice-versa. As pointed out by Titov (2011), such an assumption is likely to be too restrictive since most NLP tasks adopt word-based or lexicon-based features that vary significantly across different domains.

Regarding using neural networks for sequential labelling, our approach shares similarity with that of Collobert et al. (2011). In particular, we both use a non-linear layer to model complex relations underling word embeddings. However, our network differs from theirs in the following aspects. Collobert et al. (2011) model the dependency between neighbouring tags in a generative manner, by employing a transition score A_{ij}. Training the score involves a forward process of complexity $O(nT^2)$, where T denotes the number of tags. Our model captures such a dependency in a discriminative manner, by just adding tag-related features to the sparse-feature module. In addition, Collobert et al. (2011) train their network by maximizing the training set likelihood, while our approach is to minimize the margin loss using guided learning.

7 Conclusion

We built a web-domain POS tagger using a two-phase approach. We used a WRRBM to learn the representation of the web text and incorporate the representation in a neural network, which is trained using guided learning for easy-first POS tagging. Experiment showed that our approach achieved significant improvement in tagging the web domain text. In addition, we found that keeping the learned representations unchanged yields better performance compared with further optimizing them on the source domain data. We release our tools at https://github.com/majineu/TWeb.

For future work, we would like to investigate the two-phase approach to more challenging tasks, such as web domain syntactic parsing. We believe that high-accuracy web domain taggers and parsers would benefit a wide range of downstream tasks such as machine translation[2].

8 Acknowledgements

We would like to thank Hugo Larochelle for his advices on re-implementing WRRBM. We also thank Nan Yang, Shujie Liu and Tong Xiao for the fruitful discussions, and three anonymous reviewers for their insightful suggestions. This research was supported by the National Science Foundation of China (61272376; 61300097), the research grant T2MOE1301 from Singapore Ministry of Education (MOE) and the start-up grant SRG ISTD2012038 from SUTD.

References

Rie Ando and Tong Zhang. 2005. A high-performance semi-supervised learning method for text chunk-

[2]This work is done while the first author is visiting SUTD.

ing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 1–9, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. 2007. Greedy layer-wise training of deep networks. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 153–160. MIT Press, Cambridge, MA.

Yoshua Bengio. 2009. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127. Also published as a book. Now Publishers, 2009.

Steffen Bickel, Michael Brckner, and Tobias Scheffer. 2007. Discriminative learning for differing training and test distributions. In *Proc of ICML 2007*, pages 81–88. ACM Press.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 120–128, Sydney, Australia, July. Association for Computational Linguistics.

Michael Collins. 2002. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10*, EMNLP '02, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

George E. Dahl, Ryan P. Adams, and Hugo Larochelle. 2012. Training restricted boltzmann machines on word observations. In John Langford and Joelle Pineau, editors, *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ICML '12, pages 679–686, New York, NY, USA, July. Omnipress.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proc of ICML 2011*, pages 513–520.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 742–750, Stroudsburg, PA, USA. Association for Computational Linguistics.

Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, July.

Geoffrey E. Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8):1771–1800, August.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 264–271, Prague, Czech Republic, June. Association for Computational Linguistics.

Joseph Le Roux, Jennifer Foster, Joachim Wagner, Rasul Samad Zadeh Kaljahi, and Anton Bryl. 2012. DCU-Paris13 Systems for the SANCL 2012 Shared Task. In *Proceedings of the NAACL 2012 First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*, pages 1–4, Montréal, Canada, June.

Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. 2009a. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proc of ICML 2009*, pages 609–616.

Honglak Lee, Peter Pham, Yan Largman, and Andrew Ng. 2009b. Unsupervised feature learning for audio classification using convolutional deep belief networks. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1096–1104.

Ji Ma, Jingbo Zhu, Tong Xiao, and Nan Yang. 2013. Easy-first pos tagging and dependency parsing with beam search. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 110–114, Sofia, Bulgaria, August. Association for Computational Linguistics.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York, April. Association for Computational Linguistics.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL).

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

Barbara Plank. 2009. Structural correspondence learning for parse disambiguation. In Alex Lascarides,

Claire Gardent, and Joakim Nivre, editors, *EACL (Student Research Workshop)*, pages 37–45. The Association for Computer Linguistics.

Marc'Aurelio Ranzato, Christopher Poultney, Sumit Chopra, and Yann LeCun. 2007. Efficient learning of sparse representations with an energy-based model. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1137–1144. MIT Press, Cambridge, MA.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1988. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA.

Sandeepkumar Satpal and Sunita Sarawagi. 2007. Domain adaptation of conditional probability models via feature subsetting. In *PKDD*, volume 4702 of *Lecture Notes in Computer Science*, pages 224–235. Springer.

Libin Shen, Giorgio Satta, and Aravind Joshi. 2007. Guided learning for bidirectional sequence classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 760–767, Prague, Czech Republic, June. Association for Computational Linguistics.

Buzhou Tang, Min Jiang, and Hua Xu. 2012. Varderlibt's systems for sancl2012 shared task. In *Proceedings of the NAACL 2012 First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*, Montréal, Canada, June.

Ivan Titov. 2011. Domain adaptation by constraining inter-domain variability of latent feature representation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 62–71, Portland, Oregon, USA, June. Association for Computational Linguistics.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, Uppsala, Sweden, July. Association for Computational Linguistics.

Mengqiu Wang and Christopher D. Manning. 2013. Effect of non-linear deep architecture in sequence labeling. In *Proceedings of the 6th International Joint Conference on Natural Language Processing (IJCNLP)*.

Nan Yang, Shujie Liu, Mu Li, Ming Zhou, and Nenghai Yu. 2013. Word alignment modeling with context dependent deep neural network. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 166–175, Sofia, Bulgaria, August. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2011. Syntax-based grammaticality improvement using ccg and guided search. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1147–1157, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for Chinese word segmentation and POS tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 647–657, Seattle, Washington, USA, October. Association for Computational Linguistics.

Unsupervised Solution Post Identification from Discussion Forums

Deepak P
IBM Research - India
Bangalore, India
deepak.s.p@in.ibm.com

Karthik Visweswariah
IBM Research - India
Bangalore, India
v-karthik@in.ibm.com

Abstract

Discussion forums have evolved into a dependable source of knowledge to solve common problems. However, only a minority of the posts in discussion forums are solution posts. Identifying solution posts from discussion forums, hence, is an important research problem. In this paper, we present a technique for unsupervised solution post identification leveraging a so far unexplored textual feature, that of lexical correlations between problems and solutions. We use translation models and language models to exploit lexical correlations and solution post character respectively. Our technique is designed to not rely much on structural features such as post metadata since such features are often not uniformly available across forums. Our clustering-based iterative solution identification approach based on the EM-formulation performs favorably in an empirical evaluation, beating the only unsupervised solution identification technique from literature by a very large margin. We also show that our unsupervised technique is competitive against methods that require supervision, outperforming one such technique comfortably.

1 Introduction

Discussion forums have become a popular knowledge source for finding solutions to common problems. StackOverflow[1], a popular discussion forum for programmers is among the top-100 most visited sites globally[2]. Now, there are discussion forums for almost every major product ranging from automobiles[3] to gadgets such as those of Mac[4] or Samsung[5]. These typically start with a registered user posting a question/problem[6] to which other users respond. Typical response posts include solutions or clarification requests, whereas feedback posts form another major category of forum posts. As is the case with any community of humans, discussion forums have their share of inflammatory remarks too. Mining problem-solution pairs from discussion forums has attracted much attention from the scholarly community in the recent past. Since the first post most usually contains the problem description, identifying its solutions from among the other posts in the thread has been the focus of many recent efforts (e.g., (Gandhe et al., 2012; Hong and Davison, 2009)). Extracting problem-solution pairs from forums enables the usage of such knowledge in knowledge reuse frameworks such as case-based reasoning (Kolodner, 1992) that use problem-solution pairs as raw material. In this paper, we address the problem of unsupervised solution post identification[7] from discussion forums.

Among the first papers to address the solution identification problem was the unsupervised approach proposed by (Cong et al., 2008). It employs a graph propagation method that prioritizes posts that are (a) more similar to the problem post, (b) more similar to other posts, and (c) authored by a more authoritative user, to be labeled as solution posts. Though seen to be effective in identifying solutions from travel forums, the first two assumptions, (a) and (b), were seen to be not very

[1] http://www.stackoverflow.com

[2] http://www.alexa.com/siteinfo/stackoverflow.com

[3] http://www.cadillacforums.com/

[4] https://discussions.apple.com/

[5] http://www.galaxyforums.net/

[6] We use problem and question, as well as solution and answer interchangeably in this paper.

[7] This problem has been referred to as *answer extraction* by some papers earlier. However, we use *solution identification* to refer to the problem since *answer* and *extraction* have other connotations in the Question-Answering and Information Extraction communities respectively.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 155–164,
Baltimore, Maryland, USA, June 23-25 2014.

reliable in solution identification in other kinds of discussion boards. (Catherine et al., 2012) reports a study that illustrates that non-solution posts are, on an average, as similar to the problem as solution posts in technical forums. The second assumption (i.e., (b) above) was also not seen to be useful in discussion forums since posts that are highly similar to other posts were seen to be complaints, repetitive content being more pervasive among complaint posts than solutions (Catherine et al., 2013). Having exhausted the two obvious textual features for solution identification, subsequent approaches have largely used the presence of lexical cues signifying solution-like narrative (e.g., instructive narratives such as *"check the router for any connection issues"*) as the primary content-based feature for solution identification.

All solution identification approaches since (Cong et al., 2008) have used supervised methods that require training data in the form of labeled solution and non-solution posts. The techniques differ from one another mostly in the non-textual features that are employed in representing posts. A variety of high precision assumptions such as *solution post typically follows a problem post* (Qu and Liu, 2011), *solution posts are likely to be within the first few posts*, *solution posts are likely to have been acknowledged by the problem post author* (Catherine et al., 2012), *users with high authoritativeness are likely to author solutions* (Hong and Davison, 2009), and so on have been seen to be useful in solution identification. Being supervised methods, the above assumptions are implicitly factored in by including the appropriate feature (e.g., post position in thread) in the feature space so that the learner may learn the correlation (e.g., solution posts typically are among the first few posts) using the training data. Though such assumptions on structural features, if generic enough, may be built into unsupervised techniques to aid solution identification, the variation in availability of such features across forums limits the usage of models that rely heavily on structural features. For example, some forums employ chronological order based flattening of threads (Seo et al., 2009) making reply-to information unavailable; models that harness reply-to features would then have limited utility on identifying solutions within such flattened threads. On medical forums, privacy considerations may force forum data to be dumped without author information, making a host of author-id based features unavailable. On datasets that contain data from across forums, the model may have to be aware of the absence of certain features in subsets of the data, or be modeled using features that are available on all threads.

Our Contribution: We propose an unsupervised method for solution identification. The cornerstone of our technique is the usage of a hitherto unexplored textual feature, *lexical correlations between problems and solutions*, that is exploited along with language model based characterization of solution posts. We model the lexical correlation and solution post character using regularized translation models and unigram language models respectively. To keep our technique applicable across a large variety of forums with varying availability of non-textual features, we design it to be able to work with *minimal availability of non-textual features*. In particular, we show that by using post position as the only non-textual feature, we are able to achieve accuracies comparable to supervision-based approaches that use many structural features (Catherine et al., 2013).

2 Related Work

In this section, we provide a brief overview of previous work related to our problem. Though most of the answer/solution identification approaches proposed so far in literature are supervised methods that require a labeled training corpus, there are a few that require limited or no supervision. Table 1 provides an overview of some of the more recent solution identification techniques from literature, with a focus on some features that we wish to highlight. The common observation that most problem-solving discussion threads have a problem description in the first post has been explicitly factored into many techniques; knowing the problem/question is important for solution identification since author relations between problem and other posts provide valuable cues for solution identification. Most techniques use a variety of such features as noted in Section 1. SVMs have been the most popular method for supervised and semi-supervised learning for the task of solution identification.

Of particular interest to us are approaches that use limited or no supervision, since we focus on unsupervised solution identification in this paper.

Paper Reference	Supervision	Assumptions on Problem Position	Features other than Post Content Used	Learning Technique
(Qu and Liu, 2011)	Supervised	First Post likely to be problem	HMM assumes solution follows problem	Naive Bayes & HMM
(Ding et al., 2008)	Supervised	First Post	Post Position, Author, Context Posts	CRFs
(Kim et al., 2010)	Supervised	None	Post Position, Author, Previous Posts, Profile etc.	MaxEnt, SVM, CRF
(Hong and Davison, 2009)	Supervised	First Post	Post Position, Author, Author Authority	SVM
(Catherine et al., 2012)	Supervised	First Post	Post Position, Author, Problem Author's activities wrt Post	SVM
(Catherine et al., 2013)	Limited Supervision	First Post	Post Position/Rating, Author, Author Rating, Post Ack	SVMs & Co-Training
(Cong et al., 2008)	**Unsupervised**	None	Author, Author Authority, Relation to Problem Author	Graph Propagation
Our Method	**Unsupervised**	First Post	Post Position	Translation Models & LM

Table 1: Summary of Some Solution Identification Techniquess

The **only** unsupervised approach for the task, that from (Cong et al., 2008), uses a graph propagation method on a graph modeled using posts as vertices, and relies on the assumptions that posts that bear high similarity to the problem and other posts and those authored by authoritative users are more likely to be solution posts. Some of those assumptions, as mentioned in Section 1, were later found to be not generalizable to beyond travel forums. The semi-supervised approach presented in (Catherine et al., 2013) uses a few labeled threads to bootstrap SVM based learners which are then co-trained in an iterative fashion. In addition to various features explored in literature, they use acknowledgement modeling so that posts that have been acknowledged positively may be favored for being labeled as solutions.

We will use translation and language models in our method for solution identification. Usage of translation models for modeling the correlation between textual problems and solutions have been explored earlier starting from the answer retrieval work in (Xue et al., 2008) where new queries were conceptually expanded using the translation model to improve retrieval. Translation models were also seen to be useful in segmenting incident reports into the problem and solution parts (Deepak et al., 2012); we will use an adaptation of the generative model presented therein, for our solution extraction formulation. Entity-level translation models were recently shown to be useful in modeling correlations in QA archives (Singh, 2012).

3 Problem Definition

Let a thread $\mathcal{T}$ from a discussion forum be made up of t posts. Since we assume, much like many other earlier papers, that the first post is the problem post, the task is to identify which among the remaining $t-1$ posts are solutions. There could be multiple (most likely, different) solutions within the same thread. We may now model the thread $\mathcal{T}$ as $t-1$ post pairs, each pair having the problem post as the first element, and one of the $t-1$ remaining posts (i.e., reply posts in $\mathcal{T}$) as the second element. Let $\mathcal{C} = \{(p_1, r_1), (p_2, r_2), \ldots, (p_n, r_n)\}$ be the set of such problem-reply pairs from across threads in the discussion forum. We are interested in finding a subset $\mathcal{C}'$ of $\mathcal{C}$ such that most of the pairs in $\mathcal{C}'$ are problem-solution pairs, and most of those in $\mathcal{C} - \mathcal{C}'$ are not so. In short, we would like to find problem-solution pairs from $\mathcal{C}$ such that the F-measure[8] for solution identification is maximized.

4 Our Approach

4.1 The Correlation Assumption

Central to our approach is the assumption of lexical correlation between the problem and solution

[8] http://en.wikipedia.org/wiki/F1_score

texts. At the word level, this translates to assuming that there exist word pairs such that the presence of the first word in the problem part predicts the presence/absence of the second word in the solution part well. Though not yet harnessed for solution identification, the correlation assumption is not at all novel. Infact, the assumption that similar problems have similar solutions (of which the correlation assumption is an offshoot) forms the foundation of case-based reasoning systems (Kolodner, 1992), a kind of knowledge reuse systems that could be the natural consumers of problem-solution pairs mined from forums. The usage of translation models in QA retrieval (Xue et al., 2008; Singh, 2012) and segmentation (Deepak et al., 2012) were also motivated by the correlation assumption. We use an IBM Model 1 translation model (Brown et al., 1990) in our technique; simplistically, such a model m may be thought of as a 2-d associative array where the value $m[w_1][w_2]$ is directly related to the probability of w_1 occuring in the problem when w_2 occurs in the solution.

4.2 Generative model for Solution Posts

Consider a unigram language model $\mathcal{S}_S$ that models the lexical characteristics of solution posts, and a translation model $\mathcal{T}_S$ that models the lexical correlation between problems and solutions. Our generative model models the reply part of a (p, r) pair (in which r is a solution) as being generated from the statistical models in $\{\mathcal{S}_S, \mathcal{T}_S\}$ as follows.

- For each word w_s occuring in r,
  1. Choose $z \sim U(0, 1)$
  2. If $z \leq \lambda$, Choose $w \sim Mult(\mathcal{S}_S)$
  3. Else, Choose $w \sim Mult(\mathcal{T}_S^p)$

where $\mathcal{T}_S^p$ denotes the multionomial distribution obtained from $\mathcal{T}_S$ conditioned over the words in the post p; this is obtained by assigning each candidate solution word w a weight equal to $avg\{\mathcal{T}_S[w'][w]|w' \in p\}$, and normalizing such weights across all solution words. *In short, each solution word is assumed to be generated from the language model or the translation model (conditioned on the problem words) with a probability of λ and $1 - \lambda$ respectively, thus accounting for the correlation assumption.* The generative model above is similar to the proposal in (Deepak et al., 2012), adapted suitably for our scenario. We model non-solution posts similarly with the sole difference being that they would be sampled from the analogous models $\mathcal{S}_N$ and $\mathcal{T}_N$ that characterize behavior of non-solution posts.

Example: Consider the following illustrative example of a problem and solution post:

- *Problem:* I am unable to surf the web on the BT public wifi.
- *Solution:* Maybe, you should try disconnecting and rejoining the network.

Of the solution words above, generic words such as *try* and *should* could probably be explained by (i.e., sampled from) the solution language model, whereas *disconnect* and *rejoin* could be correlated well with *surf* and *wifi* and hence are more likely to be supported better by the translation model.

4.3 Clustering-based Approach

We propose a clustering based approach so as to cluster each of the (p, r) pairs into either the *solution* cluster or the *non-solution* cluster. The objective function that we seek to maximize is the following:

$$\sum_{(p,r)\in\mathcal{C}} \begin{cases} F((p,r),\mathcal{S}_S,\mathcal{T}_S) & \textit{if label((p,r))=S} \\ F((p,r),\mathcal{S}_N,\mathcal{T}_N) & \textit{if label((p,r))=N} \end{cases} \quad (1)$$

$F((p, r), \mathcal{S}, \mathcal{T})$ indicates the conformance of the (p, r) pair (details in Section 4.3.1) with the generative model that uses the $\mathcal{S}$ and $\mathcal{T}$ models as the language and translation models respectively. The clustering based approach labels each (p, r) pair as either solution (i.e., S) or non-solution (i.e., N). Since we do not know the models or the labelings to start with, we use an iterative approach modeled on the EM meta-algorithm (Dempster et al., 1977) involving iterations, each comprising of an E-step followed by the M-step. For simplicity and brevity, instead of deriving the EM formulation, we illustrate our approach by making an analogy with the popular K-Means clustering (MacQueen, 1967) algorithm that also uses the EM formulation and crisp assignments of data points like we do. K-Means is a clustering algorithm that clusters objects represented as multi-dimensional points into k clusters where each cluster is represented by the centroid of all its members. Each iteration in K-Means starts off with assigning each

	In K-Means	In Our Approach
Data	Multi-dimensional Points	(p, r) pairs
Cluster Model	Respective Centroid Vector	Respective $\mathcal{S}$ and $\mathcal{T}$ Models for each cluster
Initialization	Random Choice of Centroids	Models learnt using (p, r) pairs labeled using the Post Position of r
E-Step	$label(d) = \arg\min_i dist(d, centroid_i)$	$label((p, r)) = \arg\max_i F((p, r), \mathcal{S}_i, \mathcal{T}_i)$ (Sec 4.3.1), and learn solution word source probabilities (Sec 4.3.2)
M-Step	$centroid_i = avg\{d \mid label(d) = i\}$	Re-learn $\mathcal{S}_S$ and $\mathcal{T}_S$ using pairs labeled S $\mathcal{S}_N$ and $\mathcal{T}_N$ using pairs labeled N (Sec 4.3.3)
Output	The clustering of points	(p, r) pairs labeled as S

Table 2: Illustrating Our Approach wrt K-Means Clustering

data object to its nearest centroid, followed by re-computing the centroid vector based on the assignments made. The analogy with K-Means is illustrated in Table 2.

Though the analogy in Table 2 serves to provide a high-level picture of our approach, the details require further exposition. In short, our approach is a 2-way clustering algorithm that uses two pairs of models, $[\mathcal{S}_S, \mathcal{T}_S]$ and $[\mathcal{S}_N, \mathcal{T}_N]$, to model solution pairs and non-solution pairs respectively. At each iteration, the post-pairs are labeled as either solution (S) or non-solution (N) based on which pair of models they better conform to. Within the same iteration, the four models are then re-learnt using the labels and other side information. At the end of the iterations, the pairs labeled S are output as solution pairs. We describe the various details in separate subsections herein.

4.3.1 E-Step: Estimating Labels

As outlined in Table 2, each (p, r) pair would be assigned to one of the classes, solution or non-solution, based on whether it conforms better with the solution models (i.e., $\mathcal{S}_S$ & $\mathcal{T}_S$) or non-solution models ($\mathcal{S}_N$ & $\mathcal{T}_N$), as determined using the $F((p, r), \mathcal{S}, \mathcal{T})$ function, i.e.,

$$label((p, r)) = \arg\max_{i \in \{S,N\}} F((p, r), \mathcal{S}_i, \mathcal{T}_i)$$

$F(.)$ falls out of the generative model:

$$F((p, r), \mathcal{S}, \mathcal{T}) = \prod_{w \in r} \lambda \times \mathcal{S}[w] + (1 - \lambda) \times \mathcal{T}^p[w]$$

where $\mathcal{S}[w]$ denotes the probability of w from $\mathcal{S}$ and $\mathcal{T}^p[w]$ denotes the probability of w from the multinomial distribution derived from $\mathcal{T}$ conditioned over the words in p, as in Section 4.2.

4.3.2 E-Step: Estimating Reply Word Source

Since the language and translation models operate at the word level, the objective function entails that we let the models learn based on their fractional contribution of the words from the language and translation models. Thus, we estimate the proportional contribution of each word from the language and translation models too, in the E-step. The fractional contributions of the word $w \in r$ in the (p, r) pair labeled as solution (i.e., S) is as follows:

$$f^{(p,r)}_{\mathcal{S}_S}(w) = \frac{\mathcal{S}_S[w]}{\mathcal{S}_S[w] + \mathcal{T}^p_S[w]}$$

$$f^{(p,r)}_{\mathcal{T}_S}(w) = \frac{\mathcal{T}^p_S[w]}{\mathcal{S}_S[w] + \mathcal{T}^p_S[w]}$$

The fractional contributions are just the actual supports for the word w, normalized by the total contribution for the word from across the two models. Similar estimates, $f^{(p,r)}_{\mathcal{S}_N}(.)$ and $f^{(p,r)}_{\mathcal{S}_N}(.)$ are made for reply words from pairs labeled N. In our example from Section 4.2, words such as *rejoin* are likely to get higher $f^{(p,r)}_{\mathcal{T}_S}(.)$ scores due to being better correlated with problem words and consequently better supported by the translation model; those such as *try* may get higher $f^{(p,r)}_{\mathcal{S}_S}(.)$ scores.

4.3.3 M-Step: Learning Models

We use the labels and reply-word source estimates from the E-step to re-learn the language and translation models in this step. As may be obvious from the ensuing discussion, those pairs labeled as solution pairs are used to learn the $\mathcal{S}_S$ and $\mathcal{T}_S$ models and those labeled as non-solution pairs are

used to learn the models with subscript N. We let each reply word contribute as much to the respective language and translation models according to the estimates in Section 4.3.2. In our example, if the word *disconnect* is assigned a source probability of 0.9 and 0.1 for the translation and language models respectively, the virtual document-pair from (p, r) that goes into the training of the respective $\mathcal{T}$ model would assume that *disconnect* occurs in r with a frequency of 0.9; similarly, the respective $\mathcal{S}$ would account for *disconnect* with a frequency of 0.1. Though fractional word frequencies are not possible in real documents, statistical models can accomodate such fractional frequencies in a straightforward manner. The language models are learnt only over the r parts of the (p, r) pairs since they are meant to characterize reply behavior; on the other hand, translation models learn over both p and r parts to model correlation.

Regularizing the $\mathcal{T}$ models: In our formulation, the language and translation models may be seen as competing for "ownership" of reply words. Consider the post and reply vocabularies to be of sizes A and B respectively; then, the translation model would have $A \times B$ variables, whereas the unigram language model has only B variables. This gives the translation model an implicit edge due to having more parameters to tune to the data, putting the language models at a disadvantage. To level off the playing field, we use a regularization[9] operation in the learning of the translation models. The IBM Model 1 learning process uses an internal EM approach where the E-step estimates the alignment vector for each problem word; this vector indicates the distribution of alignments of the problem word across the solution words. In our example, an example alignment vector for *wifi* could be: $\{rejoin : 0.4, network : 0.4, disconnect : 0.1, \ldots\}$. Our regularization method uses a parameter τ to discard the long tail in the alignment vector by resetting entries having a value $\leq \tau$ to 0.0 followed by re-normalizing the alignment vector to add up to 1.0. Such pruning is performed at each iteration in the learning of the translation model, so that the following M-steps learn the probability matrix according to such modified alignment vectors.

The semantics of the τ parameter may be intuitively outlined. If we would like to allow alignment vectors to allow a problem word to align with upto two reply words, we would need to set τ to a value close to $0.5 (= \frac{1}{2})$; ideally though, to allow for the mass consumed by an almost inevitable long tail of very low values in the alignment vector, we would need to set it to slightly lower than 0.5, say 0.4.

[9]We use the word *regularization* in a generic sense to mean adapting models to avoid overfitting; in particular, it may be noted that we are not using popular regularization methods such as L1-regularization.

Alg. 1 ***Clustering-based Solution Identification***

Input. $\mathcal{C}$, a set of (p, r) pairs
Output. $\mathcal{C}'$, the set of identified solution pairs

Initialization
1. $\forall (p, r) \in \mathcal{C}$
2. $\quad if(r.postpos = 2)\ label((p, r)) = S$
3. $\quad else\ label((p, r)) = N$
4. Learn $\mathcal{S}_S$ & $\mathcal{T}_S$ using pairs labeled S
5. Learn $\mathcal{S}_N$ & $\mathcal{T}_N$ using pairs labeled N

EM Iterations
6. $while(not\ converged \wedge \#Iterations < 10)$
   E-Step:
7. $\quad \forall (p, r) \in \mathcal{C}$
8. $\quad label((p, r)) = \arg\max_i F((p, r), \mathcal{S}_i, \mathcal{T}_i)$
9. $\quad \forall w \in r$
10. $\quad\quad Estimate\ f^{(p,r)}_{\mathcal{S}_{label(p,r)}}(w)\ ,\ f^{(p,r)}_{\mathcal{T}_{label(p,r)}}(w)$
    M-Step:
11. $\quad$ Learn $\mathcal{S}_S$ & $\mathcal{T}_S$ from pairs labeled S using the $f^{(p,r)}_{\mathcal{S}_S}(.)\ f^{(p,r)}_{\mathcal{T}_S}(.)$ estimates
12. $\quad$ Learn $\mathcal{S}_N$ & $\mathcal{T}_N$ from pairs labeled N using the $f^{(p,r)}_{\mathcal{S}_N}(.)\ f^{(p,r)}_{\mathcal{T}_N}(.)$ estimates

Output
13. Output (p, r) pairs from $\mathcal{C}$ with $label((p, r)) = S$ as $\mathcal{C}'$

4.3.4 Initialization

K-Means clustering mostly initializes centroid vectors randomly; however, it is non-trivial to initialize the complex translation and language models randomly. Moreover, an initialization such that the $\mathcal{S}_S$ and $\mathcal{T}_S$ models favor the solution pairs more than the non-solution pairs is critical so that they may progressively lean towards modeling solution behaviour better across iterations. Towards this, we make use of a structural feature; in particular, adapting the hypothesis that solutions occur in the first N posts (Ref. (Catherine et al., 2012)), *we label the pairs that have the the reply from the second post (note that the first post is assumed to be the problem post) in the thread as a solution*

post, and all others as non-solution posts. Such an initialization along with uniform reply word source probabilities is used to learn the initial estimates of the $\mathcal{S}_S$, $\mathcal{T}_S$, $\mathcal{S}_N$ and $\mathcal{T}_N$ models to be used in the E-step for the first iteration. We will show that we are able to effectively perform solution identification using our approach by exploiting just one structural feature, the post position, as above. However, we will also show that we can exploit other features as and when available, to deliver higher accuracy clusterings.

4.3.5 Method Summary

The overall method comprising the steps that have been described is presented in Algorithm 1. The initialization using the post position (Ref. Sec 4.3.4) is illustrated in Lines 1-5, whereas the EM-iterations form Steps 6 through 12. Of these, the E-step incorporates labeling (Line 8) as described in Sec 4.3.1 and reply-word source estimation (Line 10) detailed in Sec 4.3.2. The models are then re-learnt in the M-Step (Lines 11-12) as outlined in Sec 4.3.3. At the end of the iterations that may run up to 10 times if the labelings do not stabilize earlier, the pairs labeled S are output as identified solutions (Line 13).

Time Complexity: Let n denote $|\mathcal{C}|$, and the number of unique words in each problem and reply post be a and b respectively. We will denote the vocabulary size of problem posts as A and that of reply posts as B. Learning of the language and translation models in each iteration costs $\mathcal{O}(nb + B)$ and $\mathcal{O}(k'(nab + AB))$ respectively (assuming the translation model learning runs for k' iterations). The E-step labeling and source estimation cost $\mathcal{O}(nab)$ each. For k iterations of our algorithm, this leads to an overall complexity of $\mathcal{O}(kk'(nab + AB))$.

5 Experimental Evaluation

We use a crawl of 140k threads from Apple Discussion forums[10]. Out of these, 300 threads (comprising 1440 posts) were randomly chosen and each post was manually tagged as either solution or non-solution by the authors of (Catherine et al., 2013) (who were kind enough to share the data with us) with an inter-annotator agreement[11] of 0.71. On an average, 40% of replies in each thread and 77% of first replies were seen to be solutions,

[10] http://discussions.apple.com

[11] http://en.wikipedia.org/wiki/Cohen's_kappa

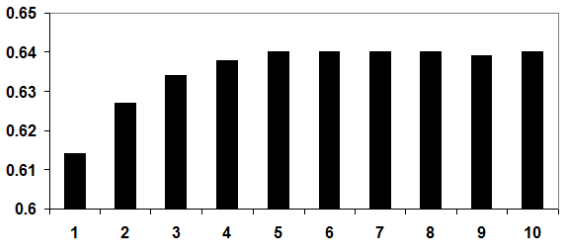


Figure 1: F% (Y) vs. #Iterations (X)

	ProblemWord, SolutionWord	$\mathcal{T}_S[p][s]$
$\mathcal{T}_S$	*network, guest*	0.0754
	connect, adaptor	0.0526
	wireless, adaptor	0.0526
	translat, shortcut	0.0492
	updat, rebuilt	0.0405
	SolutionWord	$\mathcal{S}_S[s]$
$\mathcal{S}_S$	*your*	0.0115
	try	0.0033
	router	0.0033
	see	0.0033
	password	0.0023

Table 4: Sample $\mathcal{T}_S$ and $\mathcal{S}_S$ Estimates

leading to an F-measure of 53% for our initialization heuristic. We use the F-measure[12] for solution identification, as the primary evaluation measure. While we vary the various parameters separately in order to evaluate the trends, we use a dataset of 800 threads (containing the 300 labeled threads) and set $\lambda = 0.5$ and $\tau = 0.4$ unless otherwise mentioned. Since we have only 300 labeled threads, accuracy measures are reported on those (like in (Catherine et al., 2013)). We pre-process the post data by stemming words (Porter, 1980).

5.1 Quality Evaluation

In this study, we compare the performance of our method under varying settings of λ against the only unsupervised approach for solution identification from literature, that from (Cong et al., 2008). We use an independent implementation of the technique using Kullback-Leibler Divergence (Kullback, 1997) as the similarity measure between posts; KL-Divergence was seen to perform best in the experiments reported in (Cong et al., 2008).

Table 3 illustrates the comparative performance

[12] http://en.wikipedia.org/wiki/F1_score

Technique		Precision	Recall	F-Measure
Unsupervised Graph Propagation (Cong et al., 2008)		29.7 %	55.6 %	**38.7 %**
Our Method with only Translation Models ($\lambda = 0.0$)		41.8 %	86.8 %	**56.5 %**
Our Method with only Language Models ($\lambda = 1.0$)		63.2 %	62.1 %	**62.6 %**
Our Method with Both Models ($\lambda = 0.5$)		61.3 %	66.9 %	**64.0 %**
Methods using **Supervision** (Catherine et al., 2013)	ANS CT	40.6 %	88.0 %	**55.6 %**
	ANS-ACK PCT	56.8 %	84.1 %	**67.8%**

Table 3: Quality Evaluation

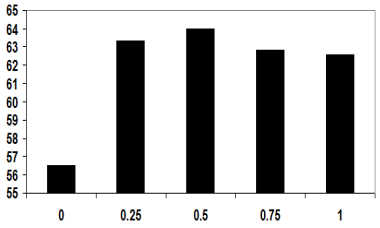


Figure 2: F% (Y) vs. λ (X)

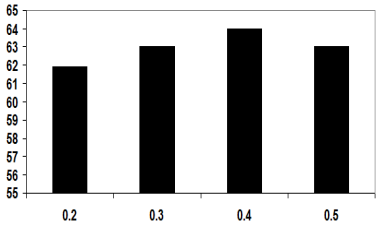


Figure 3: F% (Y) vs. τ (X)

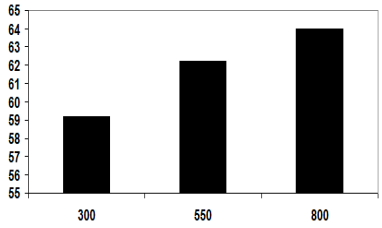


Figure 4: F% (Y) vs. #Threads (X)

on various quality metrics, of which F-Measure is typically considered most important. Our pure-LM[13] setting (i.e., $\lambda = 1$) was seen to perform up to 6 F-Measure points better than the pure-TM[14] setting (i.e., $\lambda = 0$), whereas the uniform mix is seen to be able to harness both to give a 1.4 point (i.e., 2.2%) improvement over the pure-LM case. The comparison with the approach from (Cong et al., 2008) illustrates that our method is very clearly the superior method for solution identification outperforming the former by large margins on all the evaluation measures, with the improvement on F-measure being more than 25 points.

Comparison wrt Methods from (Catherine et al., 2013): Table 3 also lists the performance of SVM-based methods from (Catherine et al., 2013) that use supervised information for solution identification, to help put the performance of our technique in perspective. Of the two methods therein, ANS CT is a more general method that uses two views (structural and lexical) of solutions which are then co-trained. ANS-ACK PCT is an enhanced method that requires author-id information and a means of classifying posts as acknowledgements (which is done using additional supervision); a post being acknowledged by the problem author is then used as a signal to enhance the solution-ness of a post. In the absence of author information (such as may be common in privacy-constrained domains such as medical forums) and extrinsic information to enable identify acknowledgements, ANS CT is the only technique available. Our technique is seen to outperform ANS CT by a respectable margin (8.6 F-measure points) while trailing behind the enhanced ANS-ACK PCT method with a reasonably narrow 3.8 F-measure point margin. Thus, our unsupervised method is seen to be a strong competitor even for techniques using supervision outlined in (Catherine et al., 2013), illustrating the effectiveness of LM and TM modeling of reply posts.

Across Iterations: For scenarios where computation is at a premium, it is useful to know how quickly the quality of solution identification stabilizes, so that the results can be collected after fewer iterations. Figure 1 plots the F-measure across iterations for the run with $\lambda = 0.5, \tau = 0.4$ setting, where the F-measure is seen to stabilize in as few as 4-5 iterations. Similar trends were observed for other runs as well, confirming that the run may be stopped as early as after the fourth iteration without considerable loss in quality.

Example Estimates from LMs and TMs: In order to understand the behavior of the statistical models, we took the highest 100 entries from both $\mathcal{S}_S$ and $\mathcal{T}_S$ and attempted to qualitatively evaluate semantics of the words (or word pairs) corresponding to those. Though the stemming made it hard to make sense of some entries, we present some of the understandable entries from among

[13]Language Model
[14]Translation Model

the top-100 in Table 4. The first three entries from $\mathcal{T}_S$ deal with connection issues for which *adaptor* or *guest account* related solutions are proposed, whereas the remaining have something to do with the *mac translator app* and *rebuilding* libraries after an *update*. The top words from $\mathcal{S}_S$ include imperative words and words from solutions to common issues that include actions pertaining to the *router* or *password*.

5.2 Varying Parameter Settings

We now analyse the performance of our approach against varying parameter settings. In particular, we vary λ and τ values and the dataset size, and experiment with some initialization variations.

Varying λ: λ is the weighting parameter that indicates the fraction of weight assigned to LMs (vis-a-vis TMs). As may be seen from Figure 2, the quality of the results as measured by the F-measure is seen to peak around the middle (i.e., $\lambda = 0.5$), and decline slowly towards either extreme, with a sharp decline at $\lambda = 0$ (i.e., pure-TM setting). This indicates that a uniform mix is favorable; however, if one were to choose only one type of model, usage of LMs is seen to be preferable than TMs.

Varying τ: τ is directly related to the extent of pruning of TMs, in the regularization operation; all values in the alignment vector $\leq \tau$ are pruned. Thus, each problem word is roughly allowed to be aligned with at most $\sim \frac{1}{\tau}$ solution words. The trends from Figure 3 suggests that allowing a problem word to be aligned to up to 2.5 solution words (i.e., $\tau = 0.4$) is seen to yield the best performance though the quality decline is graceful towards either side of the $[0.1, 0.5]$ range.

Varying Data Size: Though more data always tends to be beneficial since statistical models benefit from redundancy, the marginal utility of additional data drops to very small levels beyond a point; we are interested in the amount of data beyond which the quality of solution identification flattens out. Figure 4 suggests that there is a sharp improvement in quality while increasing the amount of data from 300 threads (i.e., 1440 (p, r) pairs) to 550 (2454 pairs), whereas the increment is smaller when adding another 250 pairs (total of 3400 pairs). Beyond 800 threads, the F-measure was seen to flatten out rapidly and stabilize at $\sim 64\%$.

Initialization: In Apple discussion forums, posts by Apple employees that are labeled with the *Apple employees* tag (approximately $\sim 7\%$ of posts in our dataset) tend to be solutions. So are posts that are marked *Helpful* ($\sim 3\%$ of posts) by other users. Being specific to Apple forums, we did not use them for initialization in experiments so far with the intent of keeping the technique generic. However, when such posts are initialized as solutions (in addition to first replies as we did earlier), the F-score for solution identification for our technique was seen to improve slightly, to 64.5% (from 64%). Thus, our technique is able to exploit any extra solution identifying structural features that are available.

6 Conclusions and Future Work

We considered the problem of unsupervised solution post identification from discussion forum threads. Towards identifying solutions to the problem posed in the initial post, we proposed the usage of a hitherto unexplored textual feature for the solution identification problem; that of lexical correlations between problems and solutions. We model and harness lexical correlations using translation models, in the company of unigram language models that are used to characterize reply posts, and formulate a clustering-based EM approach for solution identification. We show that our technique is able to effectively identify solutions using just one non-content based feature, the post position, whereas previous techniques in literature have depended heavily on structural features (that are not always available in many forums) and supervised information. Our technique is seen to outperform the sole unsupervised solution identification technique in literature, by a large margin; further, our method is even seen to be competitive to recent methods that use supervision, beating one of them comfortably, and trailing another by a narrow margin. In short, our empirical analysis illustrates the superior performance and establishes our method as the method of choice for unsupervised solution identification.

Exploration into the usage of translation models to aid other operations in discussion forums such as proactive word suggestions for solution authoring would be interesting direction for follow-up work. Discovery of problem-solution pairs in cases where the problem post is not known beforehand, would be a challenging problem to address.

References

Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Fredrick Jelinek, John D Lafferty, Robert L Mercer, and Paul S Roossin. 1990. A statistical approach to machine translation. *Computational linguistics*, 16(2):79–85.

Rose Catherine, Amit Singh, Rashmi Gangadharaiah, Dinesh Raghu, and Karthik Visweswariah. 2012. Does similarity matter? the case of answer extraction from technical discussion forums. In *COLING (Posters)*, pages 175–184.

Rose Catherine, Rashmi Gangadharaiah, Karthik Visweswariah, and Dinesh Raghu. 2013. Semi-supervised answer extraction from discussion forums. In *IJCNLP*.

Gao Cong, Long Wang, Chin-Yew Lin, Young-In Song, and Yueheng Sun. 2008. Finding question-answer pairs from online forums. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 467–474. ACM.

P. Deepak, Karthik Visweswariah, Nirmalie Wiratunga, and Sadiq Sani. 2012. Two-part segmentation of text documents. In *CIKM*, pages 793–802.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38.

Shilin Ding, Gao Cong, Chin-Yew Lin, and Xianyan Zhu. 2008. Using conditional random fields to extract contexts and answers of questions from online forums. In *ACL*.

Ankur Gandhe, Dinesh Raghu, and Rose Catherine. 2012. Domain adaptive answer extraction for discussion boards. In *Proceedings of the 21st international conference companion on World Wide Web*, pages 501–502. ACM.

Liangjie Hong and Brian D Davison. 2009. A classification-based approach to question answering in discussion boards. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 171–178. ACM.

Su Nam Kim, Li Wang, and Timothy Baldwin. 2010. Tagging and linking web forum posts. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 192–202. Association for Computational Linguistics.

Janet L Kolodner. 1992. An introduction to case-based reasoning. *Artificial Intelligence Review*, 6(1):3–34.

Solomon Kullback. 1997. *Information theory and statistics*. Courier Dover Publications.

James MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, page 14. California, USA.

Martin F Porter. 1980. An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3):130–137.

Zhonghua Qu and Yang Liu. 2011. Finding problem solving threads in online forum. In *IJCNLP*, pages 1413–1417.

Jangwon Seo, W Bruce Croft, and David A Smith. 2009. Online community search using thread structure. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1907–1910. ACM.

Amit Singh. 2012. Entity based q&a retrieval. In *EMNLP-CoNLL*, pages 1266–1277.

Xiaobing Xue, Jiwoon Jeon, and W. Bruce Croft. 2008. Retrieval models for question and answer archives. In *SIGIR*, pages 475–482.

Weakly Supervised User Profile Extraction from Twitter

Jiwei Li[1], **Alan Ritter**[2], **Eduard Hovy**[1]
[1]Language Technology Institute, [2]Machine Learning Department
Carnegie Mellon University, Pittsburgh, PA 15213, USA
bdlijiwei@gmail.com, rittera@cs.cmu.edu, ehovy@andrew.cmu.edu

Abstract

While user attribute extraction on social media has received considerable attention, existing approaches, mostly supervised, encounter great difficulty in obtaining gold standard data and are therefore limited to predicting unary predicates (e.g., gender). In this paper, we present a weakly-supervised approach to user profile extraction from Twitter. Users' profiles from social media websites such as Facebook or Google Plus are used as a distant source of supervision for extraction of their attributes from user-generated text. In addition to traditional linguistic features used in distant supervision for information extraction, our approach also takes into account network information, a unique opportunity offered by social media. We test our algorithm on three attribute domains: *spouse*, *education* and *job*; experimental results demonstrate our approach is able to make accurate predictions for users' attributes based on their tweets.[1]

1 Introduction

The overwhelming popularity of online social media creates an opportunity to display given aspects of oneself. Users' profile information in social networking websites such as Facebook[2] or Google Plus[3] provides a rich repository personal information in a structured data format, making it amenable to automatic processing. This includes, for example, users' jobs and education, and provides a useful source of information for applications such as search[4], friend recommendation, online advertising, computational social science and more.

@[shanenicholson] has taken all the kids today so I can go shopping-CHILD FREE! #iloveyoushano #iloveyoucreditcard Tamworth promo day with my handsome classy husband @[shanenicholson]
Spouse: shanenicholson
I got accepted to be part of the UofM engineering safety pilot program in [FSU] Here in class. (@ [Florida State University] - Williams Building) Don't worry , guys ! Our beloved [FSU] will always continue to rise " to the top !
Education: Florida State University (FSU)
first day of work at [HuffPo], a sports bar woo come visit me yo.. start to think we should just add a couple desks to the [HuffPo] newsroom for Business Insider writers just back from [HuffPo], what a hell !
Job: HuffPo

Table 1: Examples of Twitter message clues for user profile inference.

Although profiles exist in an easy-to-use, structured data format, they are often sparsely populated; users rarely fully complete their online profiles. Additionally, some social networking services such as Twitter don't support this type of structured profile data. It is therefore difficult to obtain a reasonably comprehensive profile of a user, or a reasonably complete facet of information (say, education level) for a class of users. While many users do not explicitly list all their personal information in their online profile, their user generated content often contains strong evidence to suggest many types of user attributes, for example education, spouse, and employment (See Table 1). Can one use such information to infer more details? In particular, can one exploit indirect clues from an unstructured data source like Twitter to obtain rich, structured user profiles?

In this paper we demonstrate that it is feasible to automatically extract Facebook-style pro-

[1]Both code and data are available at http://aclweb.org/aclwiki/index.php?title=Profile_data
[2]https://www.facebook.com/
[3]https://plus.google.com/
[4]https://www.facebook.com/about/graphsearch

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 165–174,
Baltimore, Maryland, USA, June 23-25 2014.

files directly from users' tweets, thus making user profile data available in a structured format for upstream applications. We view user profile inference as a structured prediction task where both text and network information are incorporated. Concretely, we cast user profile prediction as binary relation extraction (Brin, 1999), e.g., SPOUSE(User$_i$, User$_j$), EDUCATION(User$_i$, Entity$_j$) and EMPLOYER(User$_i$, Entity$_j$). Inspired by the concept of distant supervision, we collect training tweets by matching attribute ground truth from an outside "knowledge base" such as Facebook or Google Plus.

One contribution of the work presented here is the creation of the first large-scale dataset on three general Twitter user profile domains (i.e., EDUCATION, JOB, SPOUSE). Experiments demonstrate that by simultaneously harnessing both text features and network information, our approach is able to make accurate user profile predictions. We are optimistic that our approach can easily be applied to further user attributes such as HOBBIES and INTERESTS (MOVIES, BOOKS, SPORTS or STARS), RELIGION, HOMETOWN, LIVING LOCATION, FAMILY MEMBERS and so on, where training data can be obtained by matching ground truth retrieved from multiple types of online social media such as Facebook, Google Plus, or LinkedIn. Our contributions are as follows:

- We cast user profile prediction as an information extraction task.
- We present a large-scale dataset for this task gathered from various structured and unstructured social media sources.
- We demonstrate the benefit of jointly reasoning about users' social network structure when extracting their profiles from text.
- We experimentally demonstrate the effectiveness of our approach on 3 relations: SPOUSE, JOB and EDUCATION.

The remainder of this paper is organized as follows: We summarize related work in Section 2. The creation of our dataset is described in Section 3. The details of our model are presented in Section 4. We present experimental results in Section 5 and conclude in Section 6.

2 Related Work

While user profile inference from social media has received considerable attention (Al Zamal et al., 2012; Rao and Yarowsky, 2010; Rao et al., 2010; Rao et al., 2011), most previous work has treated this as a classification task where the goal is to predict unary predicates describing attributes of the user. Examples include gender (Ciot et al., 2013; Liu and Ruths, 2013; Liu et al., 2012), age (Rao et al., 2010), or political polarity (Pennacchiotti and Popescu, 2011; Conover et al., 2011).

A significant challenge that has limited previous efforts in this area is the lack of available training data. For example, researchers obtain training data by employing workers from Amazon Mechanical Turk to manually identify users' gender from profile pictures (Ciot et al., 2013). This approach is appropriate for attributes such as *gender* with a small numbers of possible values (e.g., *male* or *female*), for which the values can be directly identified. However for attributes such as *spouse* or *education* there are many possible values, making it impossible to manually search for gold standard answers within a large number of tweets which may or may not contain sufficient evidence.

Also related is the Twitter user timeline extraction algorithm of Li and Cardie (2013). This work is not focused on user attribute extraction, however.

Distant Supervision Distant supervision, also known as weak supervision, is a method for learning to extract relations from text using ground truth from an existing database as a source of supervision. Rather than relying on mention-level annotations, which are expensive and time consuming to generate, distant supervision leverages readily available structured data sources as a weak source of supervision for relation extraction from related text corpora (Craven et al., 1999). For example, suppose $r(e_1, e_2) = IsIn(Paris, France)$ is a ground tuple in the database and s ="Paris is the capital of France" contains synonyms for both "Paris" and "France", then we assume that s may express the fact $r(e_1, e_2)$ in some way and can be used as positive training examples. In addition to the wide use in text entity relation extraction (Mintz et al., 2009; Ritter et al., 2013; Hoffmann et al., 2011; Surdeanu et al., 2012; Takamatsu et al., 2012), distant supervision has been applied to multiple

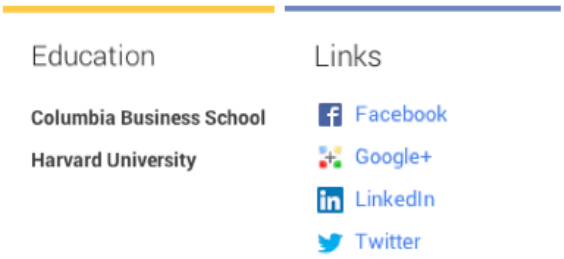


Figure 1: Illustration of Goolge Plus "knowledge base".

Figure 2: Example of fetching tweets containing entity **USC** mention from **Miranda Cosgrove** (an American actress and singer-songwriter)'s twitter stream.

fields such as protein relation extraction (Craven et al., 1999; Ravikumar et al., 2012), event extraction from Twitter (Benson et al., 2011), sentiment analysis (Go et al., 2009) and Wikipedia infobox generation (Wu and Weld, 2007).

Homophily Online social media offers a rich source of network information. McPherson et al. (2001) discovered that people sharing more attributes such as background or hobby have a higher chance of becoming friends in social media. This property, known as HOMOPHILY (summarized by the proverb "birds of a feather flock together") (Al Zamal et al., 2012) has been widely applied to community detection (Yang and Leskovec, 2013) and friend recommendation (Guy et al., 2010) on social media. In the user attribute extraction literature, researchers have considered neighborhood context to boost inference accuracy (Pennacchiotti and Popescu, 2011; Al Zamal et al., 2012), where information about the degree of their connectivity to their pre-labeled users is included in the feature vectors. A related algorithm by Mislove et al. (2010) crawled Facebook profiles of 4,000 Rice University students and alumni and inferred attributes such as major and year of matriculation purely based on network information. Mislove's work does not consider the users' text stream, however. As we demonstrate below, relying solely on network information is not enough to enable inference about attributes.

3 Dataset Creation

We now describe the generation of our distantly supervised training dataset in detail. We make use of Google Plus and Freebase to obtain ground facts and extract positive/negative bags of postings from users' twitter streams according to the ground facts.

Education/Job We first used the Google Plus API[5] (shown in Figure 1) to obtain a seed set of users whose profiles contain both their education/job status and a link to their twitter account.[6] Then, we fetched tweets containing the mention of the education/job entity from each correspondent user's twitter stream using Twitter's search API[7] (shown in Figure 2) and used them to construct positive bags of tweets expressing the associated attribute, namely EDUCATION(User$_i$, Entity$_j$), or EMPLOYER(User$_i$, Entity$_j$). The Freebase API[8] is employed for alias recognition, to match terms such as "Harvard University", "Harvard", "Harvard U" to a single The remainder of each corresponding user's entire Twitter feed is used as negative training data.[9]

We expanded our dataset from the seed users according to network information provided by Google Plus and Twitter. Concretely, we crawled circle information of users in the seed set from both their Twitter and Google Plus accounts and performed a matching to pick out shared users between one's Twitter follower list and Google Plus Circle. This process assures friend identity and avoids the problem of name ambiguity when matching accounts across websites. Among candidate users, those who explicitly display Job or Education information on Google Plus are preserved. We then gathered positive and negative data as described above.

Dataset statistics are presented in Table 2. Our

[5] https://developers.google.com/+/api/

[6] An unambiguous twitter account link is needed here because of the common phenomenon of name duplication.

[7] https://twitter.com/search

[8] http://wiki.freebase.com/wiki/Freebase_API

[9] Due to Twitter user timeline limit, we crawled at most 3200 tweets for each user.

education dataset contains 7,208 users, 6,295 of which are connected to others in the network. The positive training set for the EDUCATION is comprised of 134,060 tweets.

Spouse Facebook is the only type of social media where spouse information is commonly displayed. However, only a tiny amount of individual information is publicly accessible from Facebook Graph API[10]. To obtain ground truth for the spouse relation at large scale, we turned to Freebase[11], a large, open-domain database, and gathered instances of the /PEOPLE/PERSON/SPOUSE relation. Positive/negative training tweets are obtained in the same way as was previously described for EDUCATION and JOB. It is worth noting that our Spouse dataset is not perfect, as individuals retrieved from Freebase are mostly celebrities, and thus it's not clear whether this group of people are representative of the general population.

SPOUSE is an exception to the "homophily" effect. But it exhibits another unique property, known as, REFLEXIVITY: fact $IsSpouseOf(e_1, e_2)$ and $IsSpouseOf(e_2, e_1)$ will hold or not hold at the same time. Given training data expressing the tuple $IsSpouseOf(e_1, e_2)$ from user e_1's twitter stream, we also gather user e_2's tweet collection, and fetch tweets with the mention of e_1. We augment negative training data from e_2 as in the case of Education and Job. Our Spouse dataset contains 1,636 users, where there are 554 couples (1108 users). Note that the number of positive entities (3,121) is greater than the number of users as (1) one user can have multiple spouses at different periods of time (2) multiple entities may point to the same individual, e.g., BarackObama, Barack Obama and Barack.

4 Model

We now describe our approach to predicting user profile attributes.

4.1 Notation

Message X: Each user $i \in [1, I]$ is associated with his Twitter ID and his tweet corpus X_i. X_i is comprised of a collection of tweets $X_i = \{x_{i,j}\}_{j=1}^{j=N_i}$, where N_i denotes the number of tweets user i published.

[10] https://developers.facebook.com/docs/graph-api/

[11] http://www.freebase.com/

	Education	Job	Spouse
#Users	7,208	1,806	1,636
#Users Connected	6,295	1,407	1,108
#Edges	11,167	3,565	554
#Pos Entities	451	380	3121
#Pos Tweets	124,801	65,031	135,466
#Aver Pos Tweets per User	17.3	36.6	82.8
#Neg Entity	6,987,186	4,405,530	8,840,722
#Neg Tweets	16,150,600	10,687,403	12,872,695

Table 2: Statistics for our Dataset

Tweet Collection L_i^e: L_i^e denotes the collection of postings containing the mention of entity e from user i. $L_i^e \subset X_i$.

Entity attribute indicator $z_{i,e}^k$ and $z_{i,x}^k$: For each entity $e \in X_i$, there is a boolean variable $z_{i,e}^k$, indicating whether entity e expresses attribute k of user i. Each posting $x \in L_i^e$ is associated with attribute indicator $z_{i,x}^k$ indicating whether posting x expresses attribute k of user i. $z_{i,e}^k$ and $z_{i,x}^k$ are observed during training and latent during testing.

Neighbor set F_i^k: F_i^k denotes the neighbor set of user i. For Education $(k = 0)$ and Job $(k = 1)$, F_i^k denotes the group of users within the network that are in friend relation with user i. For Spouse attribute, F_i^k denote current user's spouse.

4.2 Model

The distant supervision assumes that if entity e corresponds to an attribute for user i, at least one posting from user i's Twitter stream containing a mention of e might express that attribute. For user-level attribute prediction, we adopt the following two strategies:

(1) GLOBAL directly makes aggregate (entity) level prediction for $z_{i,e}^k$, where features for all tweets from L_i^e are aggregated to one vector for training and testing, following Mintz et al. (2009).

(2) LOCAL makes local tweet-level predictions for each tweet $z_{i,x}^e$, $x \in L_i^k$ in the first place, making the stronger assumption that all mentions of an entity in the users' profile are expressing the associated attribute. An aggregate-level decision $z_{i,e}^k$ is then made from the deterministic OR operators.

$$z_{i,x}^e = \begin{cases} 1 & \exists x \in L_i^e, \mathbf{s.t.} z_{i,x}^k = 1 \\ 0 & \text{Otherwise} \end{cases} \quad (1)$$

The rest of this paper describes GLOBAL in detail. The model and parameters with LOCAL are identical to those in GLOBAL except that LOCAL

encode a tweet-level feature vector rather than an aggregate one. They are therefore excluded for brevity. For each attribute k, we use a model that factorizes the joint distribution as product of two distributions that separately characterize text features and network information as follows:

$$\Psi(z_{i,e}^k, X_i, F_i^k : \Theta) \propto \Psi_{text}(z_{i,e}^k, X_i)\Psi_{Neigh}(z_{i,e}^k, F_i^k) \quad (2)$$

Text Factor We use $\Psi^{text}(z_e^k, X_i)$ to capture the text related features which offer attribute clues:

$$\Psi_{text}(z_e^k, , X_i) = \exp[(\Theta_{text}^k)^T \cdot \psi_{text}(z_{i,e}^k, X_i)] \quad (3)$$

The feature vector $\psi_{text}(z_{i,e}^k, X_i)$ encodes the following standard general features:

- Entity-level: whether begins with capital letter, length of entity.
- Token-level: for each token $t \in e$, word identity, word shape, part of speech tags, name entity tags.
- Conjunctive features for a window of k (k=1,2) words and part of speech tags.
- Tweet-level: All tokens in the correspondent tweet.

In addition to general features, we employ attribute-specific features, such as whether the entity matches a bag of words observed in the list of universities, colleges and high schools for *Education* attribute, whether it matches terms in a list of companies for *Job* attribute[12]. Lists of universities and companies are taken from knowledge base NELL[13].

Neighbor Factor For Job and Education, we bias friends to have a larger possibility to share the same attribute. $\Psi_{Neigh}(z_{i,e}^k, F_i^k)$ captures such influence from friends within the network:

$$\Psi_{Neigh}(z_{i,e}^k, F_i^k) = \prod_{j \in F_i^k} \Phi_{Neigh}(z_e^k, X_j)$$

$$\Phi_{Neigh}(z_{i,e}^k, X_j) = \exp[(\Theta_{Neigh}^k)^T \cdot \psi_{Neigh}(z_{i,e}^k, X_j)] \quad (4)$$

Features we explore include the whether entity e is also the correspondent attribute with neighbor user j, i.e., $\mathbf{I}(z_{j,k}^e = 0)$ and $\mathbf{I}(z_{j,k}^e = 1)$.

[12]Freebase is employed for alias recognition.

[13]http://rtw.ml.cmu.edu/rtw/kbbrowser/

Input: Tweet Collection $\{X_i\}$, Neighbor set $\{F_i^k\}$

Initialization:

• for each user i:

for each candidate entity $e \in X_i$

$z_{i,e}^k = \text{argmax}_{z'} \Psi(z', X_i)$ from text features

End Initialization

while not convergence:

• for each user i:

update attribute values for $j \in F_i^k$

for each candidate entity $e \in X_i$

$z_{i,e}^k = \text{argmax}_{z'} \Psi(z', X_i, F_i^k)$

end while:

Figure 3: Inference for NEIGH-LATENT setting.

For Spouse, we set $F_i^{spouse} = \{e\}$ and the neighbor factor can be rewritten as:

$$\Psi_{Neigh}(z_{i,e}^k, X_j) = \Psi_{Neigh}(C_i, X_e) \quad (5)$$

It characterizes whether current user C_i to be the spouse of user e (if e corresponds to a Twitter user). We expect clues about whether C_i being entity e's spouse from e's Twitter corpus will in turn facilitate the spouse inference procedure of user i. $\psi_{Neigh}(C_i, X_e)$ encodes $\mathbf{I}(C_i \in S_e)$, $\mathbf{I}(C_i \notin S_e)$. Features we explore also include whether C_i's twitter ID appears in e's corpus.

4.3 Training

We separately trained three classifiers regarding the three attributes. All variables are observed during training; we therefore take a feature-based approach to learning structure prediction models inspired by structure compilation (Liang et al., 2008). In our setting, a subset of the features (those based on network information) are computed based on predictions that will need to be made at test time, but are observed during training. This simplified approach to learning avoids expensive inference; at test time, however, we still need to jointly predict the best attribute values for friends as is described in section 4.4.

4.4 Inference

Job and Education Our inference algorithm for Job/Education is performed on two settings, depending on whether neighbor information is

observed (NEIGH-OBSERVED) or latent (NEIGH-LATENT). Real world applications, where network information can be partly retrieved from all types of social networks, can always falls in between.

Inference in the NEIGH-OBSERVED setting is trivial; for each entity $e \in G_i$, we simply predict it's candidate attribute values using Equ.6.

$$z_{i,e}^k = \underset{z'}{\operatorname{argmax}}\, \Psi(z', X_i, F_i^k) \tag{6}$$

For NEIGH-LATENT setting, attributes for each node along the network are treated latent and user attribute prediction depends on attributes of his neighbors. The objective function for joint inference would be difficult to optimize exactly, and algorithms for doing so would be unlikely to scale to network of the size we consider. Instead, we use a sieve-based greedy search approach to inference (shown in Figure 3) inspired by recent work on coreference resolution (Raghunathan et al., 2010). Attributes are initialized using only text features, maximizing $\Psi_{text}(e, X_i)$, and ignoring network information. Then for each user we iteratively re-estimate their profile given both their text features and network features (computed based on the current predictions made for their friends) which provide additional evidence.

In this way, highly confident predictions will be made strictly from text in the first round, then the network can either support or contradict low confidence predictions as more decisions are made. This process continues until no changes are made at which point the algorithm terminates. We empirically found it to work well in practice. We expect that NEIGH-OBSERVED performs better than NEIGH-LATENT since the former benefits from gold network information.

Spouse For Spouse inference, if candidate entity e has no correspondent twitter account, we directly determine $z_{i,e}^k = \operatorname{argmax}_{z'} \Psi(z', X_i)$ from text features. Otherwise, the inference of $z_{i,e}^k$ depends on the z_{e,C_i}^k. Similarly, we initialize $z_{i,e}^k$ and z_{e,C_i}^k by maximizing text factor, as we did for Education and Job. Then we iteratively update z^k given by the rest variables until convergence.

5 Experiments

In this Section, we present our experimental results in detail.

	Education	Job
AFFINITY	74.3	14.5

Table 3: Affinity values for Education and Job.

5.1 Preprocessing and Experiment Setup

Each tweet posting is tokenized using Twitter NLP tool introduced by Noah's Ark[14] with # and @ separated following tokens. We assume that attribute values should be either name entities or terms following @ and #. Name entities are extracted using Ritter et al.'s NER system (2011). Consecutive tokens with the same named entity tag are chunked (Mintz et al., 2009). Part-of-speech tags are assigned based on Owoputi et al's tweet POS system (Owoputi et al., 2013).

Data is divided in halves. The first is used as training data and the other as testing data.

5.2 Friends with Same Attribute

Our network intuition is that users are much more likely to be friends with other users who share attributes, when compared to users who have no attributes in common. In order to statistically show this, we report the value of AFFINITY defined by Mislove et al (2010), which is used to quantitatively evaluate the degree of HOMOPHILY in the network. AFFINITY is the ratio of the fraction of links between attribute (k)-sharing users (S_k), relative to what is expected if attributes are randomly assigned in the network (E_k).

$$\begin{aligned} S_k &= \frac{\sum_i \sum_{j \in F_i^k} \mathbf{I}(P_i^k = P_j^k)}{\sum_i \sum_{j \in F_i^k} \mathbf{I}} \\ E_k &= \frac{\sum_m T_m^k (T_m^k - 1)}{U^k (U^k - 1)} \end{aligned} \tag{7}$$

where T_m^k denotes the number of users with m value for attribute k and $U^k = \sum_m T_m^k$. Table 3 shows the affinity value of the Education and Job. As we can see, the property of HOMOPHILY indeed exists among users in the social network with respect to Education and Job attribute, as significant affinity is observed. In particular, the affinity value for Education is 74.3, implying that users connected by a link in the network are 74.3 times more likely affiliated in the same school than as expected if education attributes are randomly assigned. It is interesting to note that Education exhibits a much stronger HOMOPHILY property than

[14] https://code.google.com/p/ark-tweet-nlp/downloads/list

Job. Such affinity demonstrates that our approach that tries to take advantage of network information for attribute prediction of holds promise.

5.3 Evaluation and Discussion

We evaluate settings described in Section 4.2 i.e., GLOBAL setting, where user-level attribute is predicted directly from jointly feature space and LOCAL setting where user-level prediction is made based on tweet-level prediction along with different inference approaches described in Section 4.4, i.e. NEIGH-OBSERVED and NEIGH-LATENT, regarding whether neighbor information is observed or latent.

Baselines We implement the following baselines for comparison and use identical processing techniques for each approach for fairness.

- **Only-Text:** A simplified version of our algorithm where network/neighbor influence is ignored. Classifier is trained and tested only based on text features.
- **NELL:** For Job and Education, candidate is selected as attribute value once it matches bag of words in the list of universities or companies borrowed from NELL. For Education, the list is extended by alias identification based on Freebase. For Job, we also fetch the name abbreviations[15]. NELL is only implemented for Education and Job attribute.

For each setting from each approach, we report the (P)recision, (R)ecall and (F)1-score. For LOCAL setting, we report the performance for both entity-level prediction (Entity) and posting-level prediction (Tweet). Results for Education, Job and Spouse from different approaches appear in Table 4, 5 and 6 respectively.

Local or Global For horizontal comparison, we observe that GLOBAL obtains a higher Precision score but a lower Recall than LOCAL(ENTITY). This can be explained by the fact that LOCAL(U) sets $z_{i,e}^{k} = 1$ once one posting $x \in L_i^e$ is identified as attribute related, while GLOBAL tend to be more meticulous by considering the conjunctive feature space from all postings.

Homophile effect In agreement with our expectation, NEIGH-OBSERVED performs better than NEIGH-LATENT since erroneous predictions in NEIGH-LATENT setting will have negative influence on further prediction during the greedy search process. Both NEIGH-OBSERVED and NEIGH-LATENT where network information is harnessed, perform better than **Only-Text**, which the prediction is made independently on user's text features. The improvement of NEIGH-OBSERVED over **Only-Text** is 22.7% and 6.4% regarding F-1 score for Education and Job respectively, which further illustrate the usefulness of making use of Homophile effect for attribute inference on online social media. It is also interesting to note the improvement much more significant in Education inference than Job inference. This is in accord with what we find in Section 5.2, where education network exhibits stronger HOMOPHILE property than Job network, enabling a significant benefit for education inference, but limited for job inference.

Spouse prediction also benefits from neighboring effect and the improvement is about 12% for LOCAL(ENTITY) setting. Unlike Education and Job prediction, for which in NEIGH-OBSERVED setting all neighboring variables are observed, network variables are hidden during spouse prediction. By considering network information, the model benefits from evident clues offered by tweet corpus of user e's spouse when making prediction for e, but also suffers when erroneous decision are made and then used for downstream predictions.

NELL Baseline Notably, NELL achieves highest Recall score for Education inference. It is also worth noting that most of education mentions that NELL fails to retrieve are those involve irregular spellings, such as HarvardUniv and Cornell_U, which means Recall score for NELL baseline would be even higher if these irregular spellings are recognized in a more sophisticated system. The reason for such high recall is that as our ground truths are obtained from Google plus, the users from which are mostly affiliated with decent schools found in NELL dictionary. However, the high recall from NELL is sacrificed at precision, as users can mention school entities in many of situations, such as paying a visit or reporting some relevant news. NELL will erroneously classify these cases as attribute mentions.

NELL does not work out for Job, with a fairly poor 0.0156 F1 score for LOCAL(ENTITY) and 0.163 for LOCAL(TWEET). Poor precision is expected for as users can mention firm entity in a great many of situations. The recall score for

[15] http://www.abbreviations.com/

		GLOBAL			LOCAL(ENTITY)			LOCAL(TWEET)		
		P	R	F	P	R	F	P	R	F
Our approach	NEIGH-OBSERVED	**0.804**	**0.515**	**0.628**	**0.524**	0.780	**0.627**	**0.889**	0.729	**0.801**
	NEIGH-LATENT	0.755	0.440	0.556	0.420	0.741	0.536	0.854	0.724	0.783
Only-Text	—-	0.735	0.393	0.512	0.345	0.725	0.467	0.809	0.724	0.764
NELL	—-	—-	—-	—-	0.170	**0.798**	0.280	0.616	**0.848**	0.713

Table 4: Results for Education Prediction

		GLOBAL			LOCAL(ENTITY)			LOCAL(TWEET)		
		P	R	F	P	R	F	P	R	F
Our approach	NEIGH-OBSERVED	**0.643**	**0.330**	**0.430**	**0.374**	**0.620**	**0.467**	**0.891**	**0.698**	**0.783**
	NEIGH-LATENT	0.617	0.320	0.421	0.226	0.544	0.319	0.804	0.572	0.668
Only-Text	—-	0.602	0.304	0.404	0.155	0.501	0.237	0.764	0.471	0.583
NELL	—-	—-	—-	—-	0.0079	0.509	0.0156	0.094	0.604	0.163

Table 5: Results for Job Prediction

		GLOBAL			LOCAL(ENTITY)			LOCAL(TWEET)		
		P	R	F	P	R	F	P	R	F
Our approach	—-	**0.870**	**0.560**	**0.681**	**0.593**	**0.857**	**0.701**	**0.904**	**0.782**	**0.839**
Only-Text	—-	0.852	0.448	0.587	0.521	0.781	0.625	0.890	0.729	0.801

Table 6: Results for Spouse Prediction

NELL in job inference is also quite low as job related entities exhibit a greater diversity of mentions, many of which are not covered by the NELL dictionary.

Vertical Comparison: Education, Job and Spouse Job prediction turned out to be much more difficult than Education, as shown in Tables 4 and 5. Explanations are as follows: (1) Job contains a much greater diversity of mentions than Education. Education inference can benefit a lot from the dictionary relevant feature which Job may not. (2) Education mentions are usually associated with clear evidence such as homework, exams, studies, cafeteria or books, while situations are much more complicated for job as vocabularies are usually specific for different types of jobs. (3) The boundary between a user working in and a fun for a specific operation is usually ambiguous. For example, a Google engineer may constantly update information about outcome products of Google, so does a big fun. If the aforementioned engineer barely tweets about working conditions or colleagues (which might still be ambiguous), his tweet collection, which contains many of mentions about outcomes of Google product, will be significantly similar to tweets published by a Google fun. Such nuisance can be partly solved by the consideration of network information, but not totally.

The relatively high F1 score for spouse prediction is largely caused by the great many of non-individual related entities in the dataset, the identification of which would be relatively simpler. A deeper look at the result shows that the classifier frequently makes wrong decisions for entities such as userID and name entities. Significant as some spouse relevant features are, such as love, husband, child, in most circumstances, spouse mentions are extremely hard to recognize. For example, in tweets "Check this out, @alancross, it's awesome `bit.ly/1bnjYHh`." or "Happy Birthday @alancross !". *alancross* can reasonably be any option among current user's friend, colleague, parents, child or spouse. Repeated mentions add no confidence. Although we can identify *alancross* as spouse attribute once it jointly appear with other strong spouse indicators, they are still many cases where they never co-appear. How to integrate more useful side information for spouse recognition constitutes our future work.

6 Conclusion and Future Work

In this paper, we propose a framework for user attribute inference on Twitter. We construct the publicly available dataset based on distant supervision and experiment our model on three useful user profile attributes, i.e., Education, Job and Spouse. Our model takes advantage of network information on social network. We will keep updating the dataset as more data is collected.

One direction of our future work involves exploring more general categories of user profile at-

tributes, such as interested books, movies, hometown, religion and so on. Facebook would an ideal ground truth knowledge base. Another direction involves incorporating richer feature space for better inference performance, such as multi-media sources (i.e. pictures and video).

7 Acknowledgments

A special thanks is owned to Dr. Julian McAuley and Prof. Jure Leskovec from Stanford University for the Google+ circle/network crawler, without which the network analysis would not have been conducted. This work was supported in part by DARPA under award FA8750-13-2-0005.

References

Faiyaz Zamal, Wendy Liu, and Derek Ruths. 2012. Homophily and latent attribute inference: Inferring latent attributes of twitter users from neighbors. In *ICWSM*.

Edward Benson, Aria Haghighi, and Regina Barzilay. 2011. Event discovery in social media feeds. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 389–398. Association for Computational Linguistics.

Sergey Brin. 1999. Extracting patterns and relations from the world wide web. In *The World Wide Web and Databases*.

Morgane Ciot, Morgan Sonderegger, and Derek Ruths. 2013. Gender inference of twitter users in non-english contexts. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, Wash*, pages 18–21.

Michael Conover, Jacob Ratkiewicz, Matthew Francisco, Bruno Gonçalves, Filippo Menczer, and Alessandro Flammini. 2011. Political polarization on twitter. In *ICWSM*.

Mark Craven and Johan Kumlien 1999. Constructing biological knowledge bases by extracting information from text sources. In *ISMB*, volume 1999, pages 77–86.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, pages 1–12.

Ido Guy, Naama Zwerdling, Inbal Ronen, David Carmel, and Erel Uziel. 2010. Social media recommendation based on people and tags. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 194–201. ACM.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke S Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *ACL*, pages 541–550.

Jiwei Li and Claire Cardie. 2013. Timeline generation: Tracking individuals on twitter. *Proceedings of the 23rd international conference on World wide web*.

Percy Liang, Hal Daumé III, and Dan Klein. 2008. Structure compilation: trading structure for features. In *Proceedings of the 25th international conference on Machine learning*.

Wendy Liu and Derek Ruths. 2013. Whats in a name? using first names as features for gender inference in twitter. In *2013 AAAI Spring Symposium Series*.

Wendy Liu, Faiyaz Zamal, and Derek Ruths. 2012. Using social media to infer gender composition of commuter populations. In *Proceedings of the When the City Meets the Citizen Workshop, the International Conference on Weblogs and Social Media*.

Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology*, pages 415–444.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Alan Mislove, Bimal Viswanath, Krishna Gummadi, and Peter Druschel. 2010. You are who you know: inferring user profiles in online social networks. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 251–260. ACM.

Olutobi Owoputi, Brendan OConnor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of NAACL-HLT*, pages 380–390.

Marco Pennacchiotti and Ana Popescu. 2011. A machine learning approach to twitter user classification. In *ICWSM*.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A multi-pass sieve for coreference resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*.

Delip Rao and David Yarowsky. 2010. Detecting latent user properties in social media. In *Proc. of the NIPS MLSN Workshop*.

Delip Rao, David Yarowsky, Abhishek Shreevats, and Manaswi Gupta. 2010. Classifying latent user attributes in twitter. In *Proceedings of the 2nd international workshop on Search and mining user-generated contents*, pages 37–44. ACM.

Delip Rao, Michael Paul, Clayton Fink, David Yarowsky, Timothy Oates, and Glen Coppersmith. 2011. Hierarchical bayesian models for latent attribute detection in social media. In *ICWSM*.

Haibin Liu, Michael Wall, Karin Verspoor, et al. 2012. Literature mining of protein-residue associations with graph rules learned through distant supervision. *Journal of biomedical semantics*, 3(Suppl 3):S2.

Alan Ritter, Sam Clark, Mausam, Oren Etzioni, et al. 2011. Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534. Association for Computational Linguistics.

Alan Ritter, Luke Zettlemoyer, Mausam, and Oren Etzioni. 2013. Modeling missing data in distant supervision for information extraction.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465. Association for Computational Linguistics.

Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. 2012. Reducing wrong labels in distant supervision for relation extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 721–729. Association for Computational Linguistics.

Fei Wu and Daniel S Weld. 2007. Autonomously semantifying wikipedia. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 41–50. ACM.

Jaewon Yang and Jure Leskovec. 2013. Overlapping community detection at scale: A nonnegative matrix factorization approach. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 587–596. ACM.

The effect of wording on message propagation: Topic- and author-controlled natural experiments on Twitter

Chenhao Tan
Dept. of Computer Science
Cornell University
chenhao@cs.cornell.edu

Lillian Lee
Dept. of Computer Science
Cornell University
llee@cs.cornell.edu

Bo Pang
Google Inc.
bopang42@gmail.com

Abstract

Consider a person trying to spread an important message on a social network. He/she can spend hours trying to craft the message. Does it actually matter? While there has been extensive prior work looking into predicting popularity of social-media content, the effect of wording per se has rarely been studied since it is often confounded with the popularity of the author and the topic. To control for these confounding factors, we take advantage of the surprising fact that there are many pairs of tweets containing the *same* url and written by the *same* user but employing different wording. Given such pairs, we ask: which version attracts more retweets? This turns out to be a more difficult task than predicting popular topics. Still, humans can answer this question better than chance (but far from perfectly), and the computational methods we develop can do better than both an average human and a strong competing method trained on non-controlled data.

1 Introduction

How does one make a message "successful"? This question is of interest to many entities, including political parties trying to *frame* an issue (Chong and Druckman, 2007), and individuals attempting to make a point in a group meeting. In the first case, an important type of success is achieved if the national conversation adopts the rhetoric of the party; in the latter case, if other group members repeat the originating individual's point.

The massive availability of online messages, such as posts to social media, now affords researchers new means to investigate at a very large scale the factors affecting message propagation, also known as adoption, sharing, spread, or virality. According to prior research, important features include characteristics of the originating author (e.g., verified Twitter user or not, author's messages' past success rate), the author's social network (e.g., number of followers), message timing, and message content or topic (Artzi et al., 2012; Bakshy et al., 2011; Borghol et al., 2012; Guerini et al., 2011; Guerini et al., 2012; Hansen et al., 2011; Hong et al., 2011; Lakkaraju et al., 2013; Milkman and Berger, 2012; Ma et al., 2012; Petrović et al., 2011; Romero et al., 2013; Suh et al., 2010; Sun et al., 2013; Tsur and Rappoport, 2012). Indeed, it's not surprising that one of the most retweeted tweets of all time was from user BarackObama, with 40M followers, on November 6, 2012: "Four more years. [link to photo]".

Our interest in this paper is the effect of alternative message *wording*, meaning *how* the message is said, rather than what the message is about. In contrast to the identity/social/timing/topic features mentioned above, wording is one of the few factors directly under an author's control when he or she seeks to convey a **fixed** piece of content. For example, consider a speaker at the ACL business meeting who has been tasked with proposing that Paris be the next ACL location. This person cannot on the spot become ACL president, change the shape of his/her social network, wait until the next morning to speak, or campaign for Rome instead; but he/she can craft the message to be more humorous, more informative, emphasize certain aspects instead of others, and so on. In other words, we investigate whether a different choice of words affects message propagation, *controlling for user and topic*: would user BarackObama have gotten significantly more (or fewer) retweets if he had used some alternate wording to announce his re-election?

Although we cannot create a parallel universe

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 175–185,
Baltimore, Maryland, USA, June 23-25 2014.

Table 1: Topic- and author-controlled (TAC) pairs. Topic control = inclusion of the same URL.

author	tweets	#retweets
natlsecuritycnn	t_1: FIRST ON CNN: After Petraeus scandal, Paula Broadwell looks to recapture 'normal life.' **http://t.co/qy7GGuYW**	$n_1 = 5$
	t_2: First on CNN: Broadwell photos shared with Security Clearance as she and her family fight media portrayal of her **[same URL]**	$n_2 = 29$
ABC	t_1: Workers, families take stand against Thanksgiving hours: **http://t.co/J9mQHiIEqv**	$n_1 = 46$
	t_2: Staples, Medieval Times Workers Say Opening Thanksgiving Day Crosses the Line **[same URL]**	$n_2 = 27$
cactus_music	t_1: I know at some point you've have been saved from hunger by our rolling food trucks friends. Let's help support them! **http://t.co/zg9jwA5j**	$n_1 = 2$
	t_2: Food trucks are the epitome of small independently owned LOCAL businesses! Help keep them going! Sign the petition **[same URL]**	$n_2 = 13$

in which BarackObama tweeted something else[1], fortunately, a surprising characteristic of Twitter allows us to run a fairly analogous *natural experiment*: external forces serendipitously provide an environment that resembles the desired controlled setting (DiNardo, 2008). Specifically, *it turns out to be unexpectedly common for the same user to post different tweets regarding the same URL* — a good proxy for fine-grained topic[2] — within a relatively short period of time.[3] Some example pairs are shown in Table 1; we see that the paired tweets may differ dramatically, going far beyond word-for-word substitutions, so that quite interesting changes can be studied.

Looking at these examples, can one in fact tell from the wording which tweet in a topic- and author-controlled pair will be more successful? The answer may not be a priori clear. For example, for the first pair in the table, one person we asked found t_1's invocation of a "scandal" to be more attention-grabbing; but another person preferred t_2 because it is more informative about the URL's content and includes "fight media portrayal". In an Amazon Mechanical Turk (AMT) experiment (§4), we found that humans achieved an average accuracy of 61.3%: not that high, but better than chance, indicating that it is somewhat possible for humans to predict greater message spread from different deliveries of the same information.

Buoyed by the evidence of our AMT study that wording effects exist, we then performed a battery of experiments to seek generally-applicable, non-Twitter-specific features of more successful phrasings. §5.1 applies hypothesis testing (with Bonferroni correction to ameliorate issues with multiple comparisons) to investigate the utility of features like informativeness, resemblance to headlines, and conformity to the community norm in language use. §5.2 further validates our findings via prediction experiments, including on completely fresh held-out data, used only once and after an array of standard cross-validation experiments.[4] We achieved 66.5% cross-validation accuracy and 65.6% held-out accuracy with a combination of our custom features and bag-of-words. Our classifier fared significantly better than a number of baselines, including a strong classifier trained on the most- and least-retweeted tweets that was even granted access to author and timing metadata.

2 Related work

The idea of using carefully controlled experiments to study effective communication strategies dates back at least to Hovland et al. (1953). Recent studies range from examining what characteristics of *New York Times* articles correlate with high re-sharing rates (Milkman and Berger, 2012) to looking at how differences in description affect the spread of content-controlled videos or images (Borghol et al., 2012; Lakkaraju et al., 2013). Simmons et al. (2011) examined the variation of quotes from different sources to examine how textual memes mutate as people pass them along, but did not control for author. Predicting the "success" of various texts such as novels and movie quotes has been the aim of additional prior work not already mentioned in §1 (Ashok et al., 2013; Louis and Nenkova, 2013; Danescu-Niculescu-Mizil et al., 2012; Pitler and Nenkova, 2008; McIntyre and Lapata, 2009). To our knowledge, there have been no large-scale studies exploring wording effects in a both topic- and author-controlled setting. Employing such controls, we find that predicting the more effective alternative wording is much harder than the previously well-studied problem of pre-

[1] Cf. the Music Lab "multiple universes" experiment to test the randomness of popularity (Salganik et al., 2006).

[2] Although hashtags have been used as coarse-grained topic labels in prior work, for our purposes, we have no assurance that two tweets both using, say, "#Tahrir" would be attempting to express the same message but in different words. In contrast, see the same-URL examples in Table 1.

[3] Moreover, Twitter presents tweets to a reader in strict chronological order, so that there are no algorithmic-ranking effects to compensate for in determining whether readers saw a tweet. And, Twitter accumulates retweet counts for the entire retweet cascade and displays them for the original tweet at the root of the propagation tree, so we can directly use Twitter's retweet counts to compare the entire reach of the different versions.

[4] And after crossing our fingers.

dicting popular content when author or topic can freely vary.

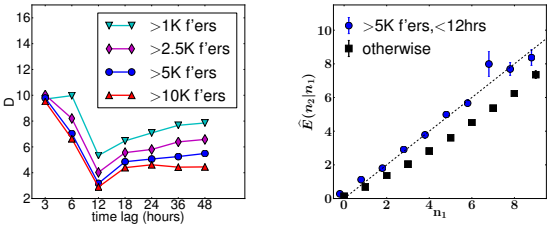
Related work regarding the features we considered is deferred to §5.1 (features description).

3 Data

Our main dataset was constructed by first gathering 1.77M topic- and author-controlled (henceforth *TAC*) tweet pairs[5] differing in more than just spacing.[6] We accomplished this by crawling timelines of 236K user ids that appear in prior work (Kwak et al., 2010; Yang and Leskovec, 2011) via the Twitter API. This crawling process also yielded 632K TAC pairs whose only difference was spacing, and an additional 558M "unpaired" tweets; as shown later in this paper, we used these extra corpora for computing language models and other auxiliary information. We applied non-obvious but important filtering — described later in this section — to control for other external factors and to reduce ambiguous cases. This brought us to a set of 11,404 pairs, with the *gold-standard* labels determined by which tweet in each pair was the one that received more retweets according to the Twitter API. We then did a second crawl to get an additional 1,770 pairs to serve as a held-out dataset. The corresponding tweet IDs are available online at `http://chenhaot.com/pages/wording-for-propagation.html`. (Twitter's terms of service prohibit sharing the actual tweets.)

Throughout, we refer to the textual content of the earlier tweet within a TAC pair as t_1, and of the later one as t_2. We denote the number of retweets received by each tweet by n_1 and n_2, respectively. We refer to the tweet with higher (lower) n_i as the "better (worse)" tweet.

Using "identical" pairs to determine how to compensate for follower-count and timing effects. In an ideal setting, differences between n_1 and n_2 would be determined solely by differences in wording. But even with a TAC pair, retweets might exhibit a temporal bias because of the chronological order of tweet presentation (t_1 might enjoy a first-mover advantage (Borghol et al., 2012) because it is the "original"; alternatively,



(a) For *identical* TAC pairs, retweet-count deviation vs. time lag between t_1 and t_2, for the author follower-counts given in the legend.

(b) Avg. n_2 vs. n_1 for identical TAC pairs, highlighting our chosen time-lag and follower thresholds. Bars: standard error. Diagonal line: $\hat{E}(n_2|n_1) = n_1$.

Figure 1: (a): The ideal case where $n_2 = n_1$ when $t_1 = t_2$ is best approximated when t_2 occurs within 12 hours of t_1 and the author has at least 10,000 or 5,000 followers. (b): in our chosen setting (blue circles), n_2 indeed tends to track n_1, whereas otherwise (black squares), there's a bias towards retweeting t_1.

t_2 might be preferred because retweeters consider t_1 to be "stale"). Also, the number of followers an author has can have complicated indirect effects on which tweets are read (space limits preclude discussion).

We use the 632K TAC pairs wherein t_1 and t_2 are *identical*[7] to check for such confounding effects: we see how much n_2 deviates from n_1 in such settings, since if wording were the only explanatory factor, the retweet rates for identical tweets ought to be equal. Figure 1(a) plots how the time lag between t_1 and t_2 and the author's follower-count affect the following deviation estimate:

$$D = \sum_{0 \leqslant n_1 < 10} |\hat{E}(n_2|n_1) - n_1|,$$

where $\hat{E}(n_2|n_1)$ is the average value of n_2 over pairs whose t_1 is retweeted n_1 times. (Note that the number of pairs whose t_1 is retweeted n_1 times decays exponentially with n_1; hence, we condition on n_1 to keep the estimate from being dominated by pairs with $n_1 = 0$, and do not consider $n_1 \geqslant 10$ because there are too few such pairs to estimate $\hat{E}(n_2|n_1)$ reliably.) Figure 1(a) shows that the setting where we (i) minimize the confounding effects of time lag and author's follower-count and (ii) maximize the amount of data to work with

[5] No data collection/processing was conducted at Google.

[6] The total excludes: tweets containing multiple URLs; tweets from users posting about the same URL more than five times (since such users might be spammers); the third, fourth, or fifth version for users posting between three and five tweets for the same URL; retweets (as identified by Twitter's API or by beginning with "RT @"); non-English tweets.

[7] Identical up to spacing: Twitter prevents exact copies by the same author appearing within a short amount of time, but some authors work around this by inserting spaces.

is: when t_2 occurs within 12 hours after t_1 and the author has more than 5,000 followers. Figure 1(b) confirms that for identical TAC pairs, our chosen setting indeed results in n_2 being on average close to n_1, which corresponds to the desired setting where wording is the dominant differentiating factor.[8]

Focus on meaningful and general changes. Even after follower-count and time-lapse filtering, we still want to focus on TAC pairs that (i) exhibit significant/interesting textual changes (as exemplified in Table 1, and as opposed to typo corrections and the like), and (ii) have n_2 and n_1 sufficiently different so that we are confident in which t_i is better at attracting retweets. To take care of (i), we discarded the 50% of pairs whose similarity was above the median, where similarity was tf-based cosine.[9] For (ii), we sorted the remaining pairs by $n_2 - n_1$ and retained only the top and bottom 5%.[10] Moreover, to ensure that we do not overfit to the idiosyncrasies of particular authors, we cap the number of pairs contributed by each author to 50 before we deal with (ii).

4 Human accuracy on TAC pairs

We first ran a pilot study on Amazon Mechanical Turk (AMT) to determine whether humans can identify, based on wording differences alone, which of two topic- and author- controlled tweets is spread more widely. Each of our 5 AMT tasks involved a disjoint set of 20 randomly-sampled TAC pairs (with t_1 and t_2 randomly reordered); subjects indicated "which tweet would other people be more likely to retweet?", provided a short justification for their binary response, and clicked a checkbox if they found that their choice was a "close call". We received 39 judgments per pair in aggregate from 106 subjects total (9 people completed all 5 tasks). The subjects' justifications were of very high quality, convincing us that they all did the task in good faith[11]. Two examples for the third TAC pair in Table 1 were: "[t_1 makes] the cause relate-able to some people, therefore showing more of an appeal as to why should they click the link and support" and, expressing the opposite view, "I like [t_2] more because [t_1] starts out with a generalization that doesn't affect me and try to make me look like I had that experience before".

If we view the set of 3900 binary judgments for our 100-TAC-pair sample as constituting independent responses, then the accuracy for this set is 62.4% (rising to 63.8% if we exclude the 587 judgments deemed "close calls"). However, if we evaluate the accuracy of the *majority* response among the 39 judgments per pair, the number rises to 73%. The accuracy of the majority response generally increases with the dominance of the majority, going above 90% when at least 80% of the judgments agree (although less than a third of the pairs satisfied this criterion).

Alternatively, we can consider the average accuracy of the 106 subjects: 61.3%, which is better than chance but far from 100%. (Variance was high: one subject achieved 85% accuracy out of 20 pairs, but eight scored below 50%.) This result is noticeably lower than the 73.8%-81.2% reported by Petrović et al. (2011), who ran a similar experiment involving two subjects and 202 tweet pairs, but where the pairs were *not* topic- or author-controlled.[12]

We conclude that even though propagation prediction becomes more challenging when topic and author controls are applied, humans can still to some degree tell which wording attracts more retweets. Interested readers can try this out themselves at `http://chenhaot.com/retweetedmore/quiz`.

5 Experiments

We now investigate computationally what wording features correspond to messages achieving a broader reach. We start (§5.1) by introducing a set of generally-applicable and (mostly) non-Twitter-specific features to capture our intuitions about what might be better ways to phrase a message. We then use hypothesis testing (§5.1) to evaluate the importance of each feature for message prop-

[8] We also computed the Pearson correlation between n_1 and n_2, even though it can be dominated by pairs with smaller n_1. The correlation is 0.853 for "$> 5K$ f'ers, <12hrs", clearly higher than the 0.305 correlation for "otherwise".

[9] Idf weighting was not employed because changes to frequent words are of potential interest. Urls, hashtags, @-mentions and numbers were normalized to [url], [hashtag], [at], and [num] before computing similarity.

[10] For our data, this meant $n_2 - n_1 \geqslant 10$ or $\leqslant -15$. Cf. our median number of retweets: 30.

[11] We also note that the feedback we got was quite positive, including: "...It's fun to make choices between close tweets and use our subjective opinion. Thanks and best of luck with your research" and "This was very interesting and really made me think about how I word my own tweets. Great job on this survey!". We only had to exclude one person (not counted among the 106 subjects), doing so because he or she gave the same uninformative justification for all pairs.

[12] The accuracy range stems from whether author's social features were supplied and which subject was considered.

Table 2: Notational conventions for tables in §5.1.

One-sided paired t-test for feature efficacy	
↑↑↑↑: p<1e-20	↓↓↓↓: p>1-1e-20
↑↑↑ : p<0.001	↓↓↓ : p>0.999
↑↑ : p<0.01	↓↓ : p>0.99
↑ : p<0.05	↓ : p>0.95
∗: passes our Bonferroni correction	

One-sided binomial test for feature increase (Do authors prefer to 'raise' the feature in t_2?)

YES : t_2 has a higher feature score than t_1, $\alpha = .05$
NO : t_2 has a lower feature score than t_1, $\alpha = .05$
(x%): $\%(f_2 > f_1)$, if sig. larger or smaller than 50%

agation and the extent to which authors employ it, followed by experiments on a prediction task (§5.2) to further examine the utility of these features.

5.1 Features: efficacy and author preference

What kind of phrasing helps message propagation? Does it work to explicitly ask people to share the message? Is it better to be short and concise or long and informative? We define an array of features to capture these and other messaging aspects. We then examine (i) how effective each feature is for attracting more retweets; and (ii) whether authors prefer applying a given feature when issuing a second version of a tweet.

First, for each feature, we use a one-sided paired t-test to test whether, on our 11K TAC pairs, our score function for that feature is larger in the better tweet versions than in the worse tweet versions, for significance levels $\alpha = .05, .01, .001,$ 1e-20. Given that we did 39 tests in total, there is a risk of obtaining false positives due to multiple testing (Dunn, 1961; Benjamini and Hochberg, 1995). To account for this, we also report significance results for the conservatively Bonferroni-corrected ("BC") significance level α = 0.05/39=1.28e-3.

Second, we examine author preference for applying a feature. We do so because one (but by no means the only) reason authors post t_2 after having already advertised the same URL in t_1 is that these authors were dissatisfied with the amount of attention t_1 got; in such cases, the changes may have been specifically intended to attract more retweets. We measure author preference for a feature by the percentage of our TAC pairs[13] where t_2 has more "occurrences" of the feature than t_1, which we denote by "$\%(f_2 > f_1)$". We use the one-sided binomial test to see whether $\%(f_2 > f_1)$ is significantly larger (or smaller) than 50%.

[13] For our preference experiments, we added in pairs where $n_2 - n_1$ was not in the top or bottom 5% (cf. §3, meaningful changes), since to measure author preference it's not necessary that the retweet counts differ significantly.

Table 3: Explicit requests for sharing (where only occurrences POS-tagged as verbs count, according to the Gimpel et al. (2011) tagger).

	effective?	author-preferred?
rt	↑↑↑↑ *	——
retweet	↑↑↑↑ *	YES (59%)
spread	↑↑↑ *	YES (56%)
please	↑↑↑ *	——
pls	↑	——
plz	↑↑	——

Table 4: Informativeness.

	effective?	author-preferred?
length (chars)	↑↑↑↑ *	YES (54%)
verb	↑↑↑↑ *	YES (56%)
noun	↑↑↑↑ *	——
adjective	↑↑↑ *	YES (51%)
adverb	↑↑↑ *	YES (55%)
proper noun	↑↑↑ *	NO (45%)
number	↑↑↑↑ *	NO (48%)
hashtag	↑	——
@-mention	↓↓↓ *	YES (53%)

Not surprisingly, it helps to ask people to share. (See Table 3; the notation for all tables is explained in Table 2.) The basic sanity check we performed here was to take as features the number of occurrences of the verbs 'rt', 'retweet', 'please', 'spread', 'pls', and 'plz' to capture explicit requests (e.g. "please retweet").

Informativeness helps. (Table 4) Messages that are more informative have increased *social exchange value* (Homans, 1958), and so may be more worth propagating. One crude approximation of informativeness is length, and we see that length helps.[14] In contrast, Simmons et al. (2011) found that shorter versions of memes are more likely to be popular. The difference may result from TAC-pair changes being more drastic than the variations that memes undergo.

A more refined informativeness measure is counts of the parts of speech that correspond to content. Our POS results, gathered using a Twitter-specific tagger (Gimpel et al., 2011), echo those of Ashok et al. (2013) who looked at predict-

[14] Of course, simply inserting garbage isn't going to lead to more retweets, but adding more information generally involves longer text.

Table 5: Conformity to the community and one's own past, measured via scores assigned by various language models.

	effective?	author-preferred?
twitter unigram	↑↑↑ *	YES (54%)
twitter bigram	↑↑↑ *	YES (52%)
personal unigram	↑↑↑ *	YES (52%)
personal bigram	———	NO (48%)

ing the success of books. The diminished effect of hashtag inclusion with respect to what has been reported previously (Suh et al., 2010; Petrović et al., 2011) presumably stems from our topic and author controls.

Be like the community, and be true to yourself (in the words you pick, but not necessarily in how you combine them). (Table 5) Although distinctive messages may attract attention, messages that conform to expectations might be more easily accepted and therefore shared. Prior work has explored this tension: Lakkaraju et al. (2013), in a content-controlled study, found that the more upvoted Reddit image titles balance novelty and familiarity; Danescu-Niculescu-Mizil et al. (2012) (henceforth DCKL'12) showed that the memorability of movie quotes corresponds to higher lexical distinctiveness but lower POS distinctiveness; and Sun et al. (2013) observed that deviating from one's own past language patterns correlates with more retweets.

Keeping in mind that the authors in our data have at least 5000 followers[15], we consider two types of language-conformity constraints an author might try to satisfy: to be similar to what is normal in the Twitter community, and to be similar to what his or her followers expect. We measure a tweet's similarity to expectations by its score according to the relevant language model, $\frac{1}{|T|}\sum_{x\in T}\log(p(x))$, where T refers to either all the unigrams (unigram model) or all and only bigrams (bigram model).[16] We trained a Twitter-community language model from our 558M unpaired tweets, and personal language models from each author's tweet history.

Imitate headlines. (Table 6) News headlines are often intentionally written to be both informative and attention-getting, so we introduce the idea of scoring by a language model built from New York Times headlines.[17]

Table 6: LM-based resemblance to headlines.

	effective?	author-preferred?
headline unigram	↑↑	YES (53%)
headline bigram	↑↑↑↑ *	YES (52%)

Table 7: Retweet score.

	effective?	author-preferred?
rt score	↑↑ *	NO (49%)
verb rt score	↑↑↑↑ *	——
noun rt score	↑↑↑ *	——
adjective rt score	↑	YES (50%)
adverb rt score	↑	YES (51%)
proper noun rt score	———	NO (48%)

Use words associated with (non-paired) retweeted tweets. (Table 7) We expect that provocative or sensationalistic tweets are likely to make people react. We found it difficult to model provocativeness directly. As a rough approximation, we check whether the changes in t_2 with respect to t_1 (which share the same topic and author) involve words or parts-of-speech that are associated with high retweet rate in a very large separate sample of unpaired tweets (retweets and replies discarded). Specifically, for each word w that appears more than 10 times, we compute the probability that tweets containing w are retweeted more than once, denoted by $rs(w)$. We define the *rt score* of a tweet as $max_{w\in T} rs(w)$, where T is all the words in the tweet, and the *rt score* of a particular POS tag z in a tweet as $max_{w\in T \& \mathrm{tag}(w)=z} rs(w)$.

Include positive and/or negative words. (Table 8) Prior work has found that including positive or negative sentiment increases message propagation (Milkman and Berger, 2012; Godes et al., 2005; Heath et al., 2001; Hansen et al., 2011). We measured the occurrence of positive and negative words as determined by the connotation lexicon of Feng et al. (2013) (better coverage than LIWC). Measuring the occurrence of both *simultaneously* was inspired by Riloff et al. (2013).

Refer to other people (but not your audience). (Table 9) First-person has been found useful for success before, but in the different domains of scientific abstracts (Guerini et al., 2012) and books (Ashok et al., 2013).

[15] This is not an artificial restriction on our set of authors; a large follower count means (in principle) that our results draw on a large sample of decisions whether to retweet or not.

[16] The tokens [at], [hashtag], [url] were ignored in the unigram-model case to prevent their undue influence, but retained in the bigram model to capture longer-range usage ("combination") patterns.

[17] To test whether the results stem from similarity to *news* rather than headlines per se, we constructed a NYT-text LM, which proved less effective. We also tried using Gawker headlines (often said to be attention-getting) but pilot studies revealed insufficient vocabulary overlap with our TAC pairs.

Table 8: Sentiment (contrast is measured by presence of both positive and negative sentiments).

	effective?	author-preferred?
positive	↑↑↑ *	——
negative	↑↑↑ *	——
contrast	↑↑↑ *	——

Table 9: Pronouns.

	effective?	author-preferred?
1st person singular	——	YES (51%)
1st person plural	——	YES (52%)
2nd person	——	YES (57%)
3rd person singular	↑↑	YES (55%)
3rd person plural	↑	YES (58%)

Generality helps. (Table 10) DCKL'12 posited that movie quotes are more shared in the culture when they are general enough to be used in multiple contexts. We hence measured the presence of indefinite articles vs. definite articles.

The easier to read, the better. (Table 11) We measure readability by using Flesch reading ease (Flesch, 1948) and Flesch-Kincaid grade level (Kincaid et al., 1975), though they are not designed for short texts. We use negative grade level so that a larger value indicates easier texts to read.

Final question: Do authors prefer to do what is effective? Recall that we use binomial tests to determine author preference for applying a feature more in t_2. Our preference statistics show that author preferences in many cases are aligned with feature efficacy. But there are several notable exceptions: for example, authors tend to increase the use of @-mentions and 2nd person pronouns even though they are ineffective. On the other hand, they did not increase the use of effective ones like proper nouns and numbers; nor did they tend to increase their rate of sentiment-bearing words. Bearing in mind that changes in t_2 may not always be intended as an effort to improve t_1, it is still interesting to observe that there are some contrasts between feature efficacy and author preferences.

5.2 Predicting the "better" wording

Here, we further examine the collective efficacy of the features introduced in §5.1 via their performance on a binary prediction task: given a TAC pair (t_1, t_2), did t_2 receive more retweets?

Our approach. We group the features introduced in §5.1 into 16 lexicon-based features (Table 3, 8, 9, 10), 9 informativeness features (Table 4), 6 language model features (Table 5, 6), 6 rt score features (Table 7), and 2 readability features (Table 11). We refer to all 39 of them together as *custom* features. We also consider tagged bag-of-words ("BOW") features, which includes all the unigram (word:POS pair) and bigram features that appear more than 10 times in the cross-validation data. This yields 3,568 unigram features and 4,095 bigram features, for a total of 7,663 so-called *1,2-gram features*. Values for each feature are normalized by linear transformation across all tweets in the training data to lie in the range $[0, 1]$.[18]

Table 10: Generality.

	effective?	author-preferred?
indefinite articles (a,an)	↑↑↑ *	——
definite articles (the)	——	YES (52%)

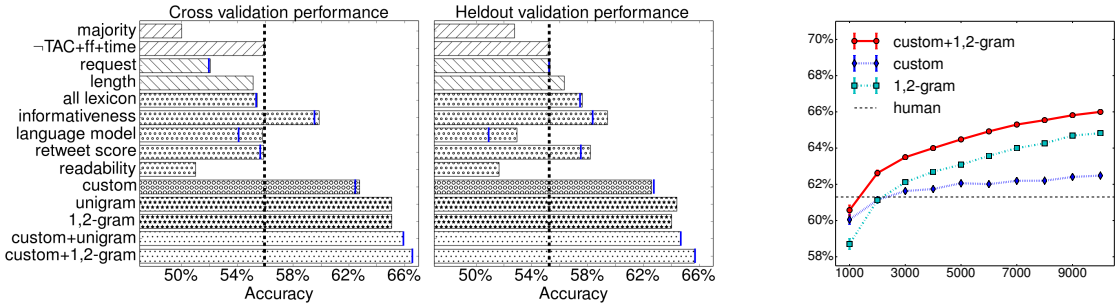
Table 11: Readability.

	effective?	author-preferred?
reading ease	↑↑	YES (52%)
negative grade level	↑	YES (52%)

For a given TAC pair, we construct its feature vector as follows. For each feature being considered, we compute its normalized value for each tweet in the pair and take the difference as the feature value for this pair. We use L2-regularized logistic regression as our classifier, with parameters chosen by cross validation on the training data. (We also experimented with SVMs. The performance was very close, but mostly slightly lower.)

A strong non-TAC alternative, with social information and timing thrown in. One baseline result we would like to establish is whether the topic and author controls we have argued for, while intuitively compelling for the purposes of trying to determine the best way for a given author to present some fixed content, are really necessary in practice. To test this, we consider an alternative binary L2-regularized logistic-regression classifier that is trained on unpaired data, specifically, on the collection of 10,000 most retweeted tweets (gold-standard label: positive) plus the 10,000 least retweeted tweets (gold-standard label: negative) that are neither retweets nor replies. Note that this alternative thus is granted, by design, roughly *twice* the training instances that our classifiers have, as a result of having roughly the same number of tweets, since our instances are pairs. Moreover, we additionally include the tweet author's follower count, and the day and hour of posting, as features. We refer to this alternative classifier as $\neg$*TAC+ff+time*. (Mnemonic: "ff" is used in bibliographic contexts as an abbreviation

[18] We also tried normalization by *whitening*, but it did not lead to further improvements.



(a) Cross-validation and heldout accuracy for various feature sets. Blue lines inside bars: performance when custom features are restricted to those that pass our Bonferroni correction (no line for readability because no readability features passed). Dashed vertical line: ¬TAC+ff+time performance.

(b) Cross-validation accuracy vs data size. Human performance was estimated from a disjoint set of 100 pairs (see §4).

Figure 2: Accuracy results. Pertinent significance results are as follows. In cross-validation, custom+1,2-gram is significantly better than ¬TAC+ff+time (p=0) and 1,2-gram (p=3.8e-7). In heldout validation, custom+1,2-gram is significantly better than ¬TAC+ff+time (p=3.4e-12) and 1,2-gram (p=0.01) but not unigram (p=0.08), perhaps due to the small size of the heldout set.

for "and the following".) We apply it to a tweet pair by computing whether it gives a higher score to t_2 or not.

Baselines. To sanity-check whether our classifier provides any improvement over the simplest methods one could try, we also report the performance of the majority baseline, our request-for-sharing features, and our character-length feature.

Performance comparison. We compare the accuracy (percentage of pairs whose labels were correctly predicted) of our approach against the competing methods. We report 5-fold cross validation results on our balanced set of 11,404 TAC pairs and on our completely disjoint heldout data[19] of 1,770 TAC pairs; this set was never examined during development, and there are no authors in common between the two testing sets.

Figure 2(a) summarizes the main results. While ¬TAC+ff+time outperforms the majority baseline, using all the features we proposed beats ¬TAC+ff+time by more than 10% in both cross-validation (66.5% vs 55.9%) and heldout validation (65.6% vs 55.3%). We outperform the average human accuracy of 61% reported in our Amazon Mechanical Turk experiments (for a different data sample); ¬TAC+ff+time fails to do so.

The importance of topic and author control can be seen by further investigation of ¬TAC+ff+time's performance. First, note that it yields an accuracy of around 55% on our alternate-version-selection task,[20] even though its cross-validation accuracy on the larger most- and least-retweeted unpaired tweets averages out to a high 98.8%. Furthermore, note the superior performance of unigrams trained on TAC data vs ¬TAC+ff+time — which is similar to our unigrams but trained on a larger but non-TAC dataset that included metadata. Thus, TAC pairs are a useful data source even for non-custom features. (We also include individual feature comparisons later.)

Informativeness is the best-performing custom feature group when run in isolation, and outperforms all baselines, as well as ¬TAC+ff+time; and we can see from Figure 2(a) that this is not due just to length. The combination of all our 39 custom features yields approximately 63% accuracy in both testing settings, significantly outperforming informativeness alone ($p<0.001$ in both cases). Again, this is higher than our estimate of average human performance.

Not surprisingly, the TAC-trained BOW features (unigram and 1,2-gram) show impressive predictive power in this task: many of our custom features can be captured by bag-of-word features, in a way. Still, the best performance is achieved

[19] To construct this data, we used the same criteria as in §3: written by authors with more than 5000 followers, posted within 12 hours, $n_2 - n_1 \geqslant 10$ or $\leqslant -15$, and cosine similarity threshold value the same as in §3, cap of 50 on number of pairs from any individual author.

[20] One might suspect that the problem is that ¬TAC+ff+time learns from its training data to overrely on follower-count, since that is presumably a good feature for non-TAC tweets, and for this reason suffers when run on TAC data where follower-counts are by construction non-informative. But in fact, we found that removing the follower-count feature from ¬TAC+ff+time and re-training did not lead to improved performance. Hence, it seems that it is the non-controlled nature of the alternate training data that explains the drop in performance.

by combining our custom and 1,2-gram features together, to a degree statistically significantly better than using 1,2-gram features alone.

Finally, we remark on our Bonferroni correction. Recall that the intent of applying it is to avoid false positives. However, in our case, Figure 2(a) shows that our potentially "false" positives — features whose effectiveness did not pass the Bonferroni correction test — actually do raise performance in our prediction tests.

Size of training data. Another interesting observation is how performance varies with data size. For $n = 1000, 2000, \ldots, 10000$, we randomly sampled n pairs from our 11,404 pairs, and computed the average cross-validation accuracy on the sampled data. Figure 2(b) shows the averages over 50 runs of the aforementioned procedure. Our custom features can achieve good performance with little data, in the sense that for sample size 1000, they outperform BOW features; on the other hand, BOW features quickly surpass them. Across the board, the custom+1,2-gram features are consistently better than the 1,2-gram features alone.

Top features. Finally, we examine some of the top-weighted individual features from our approach and from the competing ¬TAC+ff+time classifier. The top three rows of Table 12 show the best custom and best and worst unigram features for our method; the bottom two rows show the best and worst unigrams for ¬TAC+ff+time. Among custom features, we see that community and personal language models, informativeness, retweet scores, sentiment, and generality are represented. As for unigram features, not surprisingly, "rt" and "retweet" are top features for both our approach and ¬TAC+ff+time. However, the other unigrams for the two methods seem to be a bit different in spirit. Some of the unigrams determined to be most poor only by our method appear to be both surprising and yet plausible in retrospect: "icymi" (abbreviation for "in case you missed it") tends to indicate a direct repetition of older information, so people might prefer to retweet the earlier version; "thanks" and "sorry" could correspond to personal thank-yous and apologies not meant to be shared with a broader audience, and similarly @-mentioning another user may indicate a tweet intended only for that person. The appearance of [hashtag] in the best ¬TAC+ff+time unigrams is consistent with prior research in non-TAC settings (Suh et al., 2010; Petrović et al., 2011).

Table 12: Features with largest coefficients, delimited by commas. POS tags omitted for clarity.

Our approach	
best 15 custom	twitter bigram, length (chars), rt (the word), retweet (the word), verb, verb retweet score, personal unigram, proper noun, number, noun, positive words, please (the word), proper noun retweet score, indefinite articles (a,an), adjective
best 20 unigrams	rt, retweet, [num], breaking, is, win, never, ., people, need, official, officially, are, please, november, world, girl, !!!, god, new
worst 20 unigrams	:, [at], icymi, also, comments, half, ?, earlier, thanks, sorry, highlights, bit, point, update, last, helping, peek, what, haven't, debate
¬TAC+ff+time	
best 20 unigrams	[hashtag], teen, fans, retweet, sale, usa, women, butt, caught, visit, background, upcoming, rt, this, bieber, these, each, chat, houston, book
worst 20 unigrams	:, ..., boss, foundation, ?, ∼, others, john, roll, ride, appreciate, page, drive, correct, full, ', looks, @ (not as [at]), sales, hurts

6 Conclusion

In this work, we conducted the first large-scale topic- and author-controlled experiment to study the effects of wording on information propagation.

The features we developed to choose the better of two alternative wordings posted better performance than that of all our comparison algorithms, including one given access to author and timing features but trained on non-TAC data, and also bested our estimate of average human performance. According to our hypothesis tests, helpful wording heuristics include adding more information, making one's language align with both community norms and with one's prior messages, and mimicking news headlines. Readers may try out their own alternate phrasings at `http://chenhaot.com/retweetedmore/` to see what a simplified version of our classifier predicts.

In future work, it will be interesting to examine how these features generalize to longer and more extensive arguments. Moreover, understanding the underlying psychological and cultural mechanisms that establish the effectiveness of these features is a fundamental problem of interest.

Acknowledgments. We thank C. Callison-Burch, C. Danescu-Niculescu-Mizil, J. Kleinberg, P. Mahdabi, S. Mullainathan, F. Pereira, K. Raman, A. Swaminathan, the Cornell NLP seminar participants and the reviewers for their comments; J. Leskovec for providing some initial data; and the anonymous annotators for all their labeling help. This work was supported in part by NSF grant IIS-0910664 and a Google Research Grant.

References

Yoav Artzi, Patrick Pantel, and Michael Gamon. 2012. Predicting responses to microblog posts. In *Proceedings of NAACL (short paper)*.

Vikas Ganjigunte Ashok, Song Feng, and Yejin Choi. 2013. Success with style: Using writing style to predict the success of novels. In *Proceedings of EMNLP*.

Eitan Bakshy, Jake M. Hofman, Winter A. Mason, and Duncan J. Watts. 2011. Everyone's an influencer: Quantifying influence on twitter. In *Proceedings of WSDM*.

Yoav Benjamini and Yosef Hochberg. 1995. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 289–300.

Youmna Borghol, Sebastien Ardon, Niklas Carlsson, Derek Eager, and Anirban Mahanti. 2012. The untold story of the clones: Content-agnostic factors that impact YouTube video popularity. In *Proceedings of KDD*.

Dennis Chong and James N. Druckman. 2007. Framing theory. *Annual Review of Political Science*, 10:103–126.

Cristian Danescu-Niculescu-Mizil, Justin Cheng, Jon Kleinberg, and Lillian Lee. 2012. You had me at hello: How phrasing affects memorability. In *Proceedings of ACL*.

John DiNardo. 2008. Natural experiments and quasi-natural experiments. In *The New Palgrave Dictionary of Economics*. Palgrave Macmillan.

Olive Jean Dunn. 1961. Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293):52–64.

Song Feng, Jun Seok Kang, Polina Kuznetsova, and Yejin Choi. 2013. Connotation lexicon: A dash of sentiment beneath the surface meaning. In *Proceedings of ACL*.

Rudolph Flesch. 1948. A new readability yardstick. *Journal of applied psychology*, 32(3):221.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech Tagging for Twitter: Annotation, Features, and Experiments. In *Proceedings of NAACL (short paper)*.

David Godes, Dina Mayzlin, Yubo Chen, Sanjiv Das, Chrysanthos Dellarocas, Bruce Pfeiffer, Barak Libai, Subrata Sen, Mengze Shi, and Peeter Verlegh. 2005. The firm's management of social interactions. *Marketing Letters*, 16(3-4):415–428.

Marco Guerini, Carlo Strapparava, and Gözde Özbal. 2011. Exploring text virality in social networks. In *Proceedings of ICWSM (poster)*.

Marco Guerini, Alberto Pepe, and Bruno Lepri. 2012. Do linguistic style and readability of scientific abstracts affect their virality? In *Proceedings of ICWSM (poster)*.

Lars Kai Hansen, Adam Arvidsson, Finn Årup Nielsen, Elanor Colleoni, and Michael Etter. 2011. Good friends, bad news-affect and virality in Twitter. *Communications in Computer and Information Science*, 185:34–43.

Chip Heath, Chris Bell, and Emily Sternberg. 2001. Emotional selection in memes: The case of urban legends. *Journal of personality and social psychology*, 81(6):1028.

George C. Homans. 1958. Social Behavior as Exchange. *American Journal of Sociology*, 63(6):597–606.

Liangjie Hong, Ovidiu Dan, and Brian D. Davison. 2011. Predicting popular messages in Twitter. In *Proceedings of WWW*.

Carl I. Hovland, Irving L. Janis, and Harold H. Kelley. 1953. *Communication and Persuasion: Psychological Studies of Opinion Change*, volume 19. Yale University Press.

J. Peter Kincaid, Robert P. Fishburne Jr., Richard L. Rogers, and Brad S. Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, DTIC Document.

Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. 2010. What is Twitter, a social network or a news media? In *Proceedings of WWW*.

Himabindu Lakkaraju, Julian McAuley, and Jure Leskovec. 2013. What's in a name? Understanding the interplay between titles, content, and communities in social media. In *Proceedings of ICWSM*.

Annie Louis and Ani Nenkova. 2013. What makes writing great? First experiments on article quality prediction in the science journalism domain. *Transactions of ACL*.

Zongyang Ma, Aixin Sun, and Gao Cong. 2012. Will this #hashtag be popular tomorrow? In *Proceedings of SIGIR*.

Neil McIntyre and Mirella Lapata. 2009. Learning to tell tales: A data-driven approach to story generation. In *Proceedings of ACL-IJCNLP*.

Katherine L Milkman and Jonah Berger. 2012. What makes online content viral? *Journal of Marketing Research*, 49(2):192–205.

Saša Petrović, Miles Osborne, and Victor Lavrenko. 2011. RT to win! Predicting message propagation in Twitter. In *Proceedings of ICWSM*.

Emily Pitler and Ani Nenkova. 2008. Revisiting readability: A unified framework for predicting text quality. In *Proceedings of EMNLP*.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of EMNLP*.

Daniel M. Romero, Chenhao Tan, and Johan Ugander. 2013. On the interplay between social and topical structure. In *Proceedings of ICWSM*.

Matthew J. Salganik, Peter Sheridan Dodds, and Duncan J. Watts. 2006. Experimental study of inequality and unpredictability in an artificial cultural market. *Science*, 311(5762):854–856.

Matthew P. Simmons, Lada A Adamic, and Eytan Adar. 2011. Memes online: Extracted, subtracted, injected, and recollected. In *Proceedings of ICWSM*.

Bongwon Suh, Lichan Hong, Peter Pirolli, and Ed H. Chi. 2010. Want to be retweeted? Large scale analytics on factors impacting retweet in Twitter network. In *Proceedings of SocialCom*.

Tao Sun, Ming Zhang, and Qiaozhu Mei. 2013. Unexpected relevance: An empirical study of serendipity in retweets. In *Proceedings of ICWSM*.

Oren Tsur and Ari Rappoport. 2012. What's in a hashtag?: Content based prediction of the spread of ideas in microblogging communities. In *Proceedings of WSDM*.

Jaewon Yang and Jure Leskovec. 2011. Patterns of temporal variation in online media. In *Proceedings of WSDM*.

Inferring User Political Preferences from Streaming Communications

Svitlana Volkova,[1] **Glen Coppersmith**[2] **and Benjamin Van Durme**[1,2]
[1]Center for Language and Speech Processing,
[2]Human Language Technology Center of Excellence,
Johns Hopkins University, Baltimore, MD 21218
svitlana@jhu.edu, coppersmith@jhu.edu, vandurme@cs.jhu.edu

Abstract

Existing models for social media personal analytics assume access to thousands of messages per user, even though most users author content only sporadically over time. Given this sparsity, we: (i) leverage content from the local neighborhood of a user; (ii) evaluate batch models as a function of size and the amount of messages in various types of neighborhoods; and (iii) estimate the amount of time and tweets required for a dynamic model to predict user preferences. We show that even when limited or no self-authored data is available, language from friend, retweet and user mention communications provide sufficient evidence for prediction. When updating models over time based on Twitter, we find that political preference can be often be predicted using roughly 100 tweets, depending on the context of user selection, where this could mean hours, or weeks, based on the author's tweeting frequency.

1 Introduction

Inferring latent user attributes such as gender, age, and political preferences (Rao et al., 2011; Zamal et al., 2012; Cohen and Ruths, 2013) automatically from personal communications and social media including emails, blog posts or public discussions has become increasingly popular with the web getting more social and volume of data available. Resources like Twitter[1] or Facebook[2] become extremely valuable for studying the underlying properties of such informal communications because of its volume, dynamic nature, and diverse population (Lunden, 2012; Smith, 2013).

[1]http://www.demographicspro.com/
[2]http://www.wolframalpha.com/facebook/

The existing batch models for predicting latent user attributes rely on thousands of tweets per author (Rao et al., 2010; Conover et al., 2011; Pennacchiotti and Popescu, 2011a; Burger et al., 2011; Zamal et al., 2012; Nguyen et al., 2013). However, most Twitter users are less prolific than those examined in these works, and thus do not produce the thousands of tweets required to obtain their levels of accuracy e.g., the median number of tweets produced by a random Twitter user per day is 10. Moreover, recent changes to Twitter API querying rates further restrict the speed of access to this resource, effectively reducing the amount of data that can be collected in a given time period.

In this paper we analyze and go beyond static models formulating personal analytics in social media as a streaming task. We first evaluate batch models that are cognizant of low-resource prediction setting described above, maximizing the efficiency of content in calculating personal analytics. To the best of our knowledge, this is the first work that makes explicit the tradeoff between accuracy and cost (manifest as calls to the Twitter API), and optimizes to a different tradeoff than state-of-the-art approaches, seeking maximal performance when limited data is available. In addition, we propose streaming models for personal analytics that dynamically update user labels based on their stream of communications which has been addressed previously by Van Durme (2012b). Such models better capture the real-time nature of evidence being used in latent author attribute predictions tasks. Our main contributions include:

- develop low-resource and real-time dynamic approaches for personal analytics using as an example the prediction of political preference of Twitter users;
- examine the relative utility of six different notions of "similarity" between users in an implicit Twitter social network for personal analytics;

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 186–196,
Baltimore, Maryland, USA, June 23-25 2014.

- experiments are performed across multiple datasets supporting the prediction of political preference in Twitter, to highlight the significant differences in performance that arise from the underlying collection and annotation strategies.

2 Identifying Twitter Social Graph

Twitter users interact with one another and engage in direct communication in different ways e.g., using retweets, user mentions e.g., *@youtube* or hashtags e.g., *#tcot*, in addition to having explicit connections among themselves such as following, friending. To investigate all types of social relationships between Twitter users and construct Twitter social graphs we collect lists of followers and friends, and extract user mentions, hashtags, replies and retweets from communications.[3]

2.1 Social Graph Definition

Lets define an attributed, undirected graph $G = (V, E)$, where V is a set of vertices and E is a set of edges. Each vertex v_i represents someone in a communication graph i.e., *communicant*: here a Twitter user. Each vertex is attributed with a feature vector $\vec{f}(v_i)$ which encodes communications e.g., tweets available for a given user. Each vertex is associated with a latent attribute $a(v_i)$, in our case it is binary $a(v_i) \in \{D, R\}$, where D stands for Democratic and R for Republican users. Each edge $e_{ij} \in E$ represents a connection between v_i and v_j, $e_{ij} = (v_i, v_j)$ and defines different social circles between Twitter users e.g., follower (f), friend (b), user mention (m), hashtag (h), reply (y) and retweet (w). Thus, $E \in V^{(2)} \times \{f, b, h, m, w, y\}$. We denote a set of edges of a given type as $\phi_r(E)$ for $r \in \{f, b, h, m, w, y\}$. We denote a set of vertices adjacent to v_i by social circle type r as $N_r(v_i)$ which is equivalent to $\{v_j \mid e_{ij} \in \phi_r(E)\}$. Following Filippova (2012) we refer to $N_r(v_i)$ as v_i's social circle, otherwise known as a neighborhood. In most cases, we only work with a sample of a social circle, denoted by $N'_r(v_i)$ where $|N'_r(v_i)| = k$ is its size for v_i.

Figure 1 presents an example of a social graph derived from Twitter. Notably, users from different social circles can be shared across the users of the same or different classes e.g., a user v_j can be

[3]The code and detailed explanation on how we collected all six types of user neighbors and their communications using Twitter API can be found here: http://www.cs.jhu.edu/ svitlana/

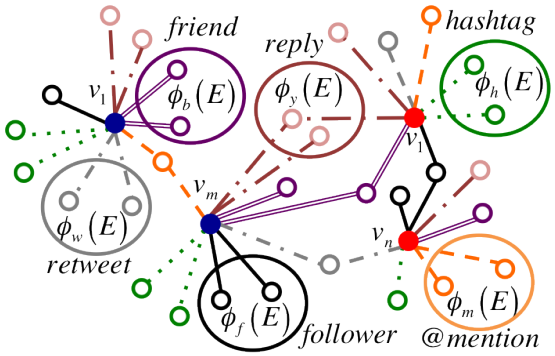


Figure 1: An example of a social graph with follower, friend, @mention, reply, retweet and hashtag social circles for each user of interest e.g., blue: Democratic, red: Republican.

in both follower circle $v_j \in N_f(v_i), v_i \in D$ and retweet circle $v_j \in N_w(v_k), v_k \in R$.

2.2 Candidate-Centric Graph

We construct candidate-centric graph G_{cand} by looking into following relationships between the users and Democratic or Republican candidates during the 2012 US Presidential election. In the Fall of 2012, leading up to the elections, we randomly sampled $n = 516$ Democratic and $m = 515$ Republican users. We labeled users as Democratic if they exclusively follow both Democratic candidates[4] – BarackObama and JoeBiden but do not follow both Republican candidates – MittRomney and RepPaulRyan and vice versa. We collectively refer to D and R as our "users of interest" for which we aim to predict political preference. For each such user we collect recent tweets and randomly sample their immediate $k = 10$ neighbors from follower, friend, user mention, reply, retweet and hashtag social circles.

2.3 Geo-Centric Graph

We construct a geo-centric graph G_{geo} by collecting $n = 135$ Democratic and $m = 135$ Republican users from the Maryland, Virginia and Delaware region of the US with self-reported political preference in their biographies. Similar to the candidate-centric graph, for each user we collect recent tweets and randomly sample user social circles in the Fall of 2012. We collect this data to get a sample of politically less active users compared to the users from candidate-centric graph.

2.4 ZLR Graph

We also consider a G_{ZLR} graph constructed from a dataset previously used for political affiliation

[4]As of Oct 12, 2012, the number of followers for Obama, Biden, Romney and Ryan were 2m, 168k, 1.3m and 267k.

classification (Zamal et al., 2012). This dataset consists of 200 Republican and 200 Democratic users associated with 925 tweets on average per user.[5] Each user has on average 6155 friends with 642 tweets per friend. Sharing restrictions and rate limits on Twitter data collection only allowed us to recreate a semblance of ZLR data[6] – 193 Democratic and 178 Republican users with 1K tweets per user, and 20 neighbors of four types including follower, friends, user mention and retweet with 200 tweets per neighbor for each user of interest.

3 Batch Models

Baseline User Model As input we are given a set of vertices representing users of interest $v_i \in V$ along with feature vectors $\vec{f}(v_i)$ derived from content authored by the user of interest. Each user is associated with a non-zero number of publicly posted tweets. Our goal is assign to a category each user of interest v_i based on $\vec{f}(v_i)$. Here we focus on a binary assignment into the categories Democratic D or Republican R. The log-linear model[7] for such binary classification is:

$$\Phi_{v_i} = \begin{cases} D & (1+\exp[-\vec{\theta}\cdot\vec{f}(v_i)])^{-1} \geq 0.5, \\ R & \text{otherwise.} \end{cases} \tag{1}$$

where features are normalized word ngram counts extracted from v_i's tweets $\vec{f}_t(v_i) : D \times t(v_i) \rightarrow \mathbb{R}$.

The proposed baseline model follows the same trends as the existing state-of-the-art approaches for user attribute classification in social media as described in Section 8. Next we propose to extend the baseline model by taking advantage of language in user social circles as describe below.

Neighbor Model As input we are given user-local neighborhood $N_r(v_i)$, where r is a neighborhood type. Besides the neighborhood's type r, each is characterized by:

- the number of communications per neighbor $\vec{f}_t(N_r)$, $t = \{5, 10, 15, 25, 50, 100, 200\}$;
- the order of the social circle – the number of neighbors per user of interest $|N_r| = deg(v_i)$, $n = \{1, 2, 5, 10\}$.

Our goal is to classify users of interest using evidence (e.g., communications) from their local neighborhood $\sum_n \vec{f}_t[N_r(v_i)] \equiv \vec{f}(N_r)$ as Democratic or Republican. The corresponding log-linear model is defined as:

$$\Phi_{N_r} = \begin{cases} D & (1+\exp[-\vec{\theta}\cdot\vec{f}(N_r)])^{-1} \geq 0.5, \\ R & \text{otherwise.} \end{cases} \tag{2}$$

To check whether our static models are cognizant of low-resource prediction settings we compare the performance of the user model from Eq.1 and the neighborhood model from Eq.2. Following the streaming nature of social media, we see the scarce available resource as the number of requests allowed per day to the Twitter API. Here we abstract this to a model assumption where we receive one tweet t_k at a time and aim to maximize classification performance with as few tweets per user as possible:[8]

- for the baseline user model:

$$\underset{k}{\text{minimize}} \quad \sum_k t_k(v_i), \tag{3}$$

- for the neighborhood model:

$$\underset{k}{\text{minimize}} \quad \sum_n \sum_k t_k[N_r(v_i)]. \tag{4}$$

4 Streaming Models

We rely on straightforward Bayesian rule update to our batch models in order to simulate a real-time streaming prediction scenario as a first step beyond the existing models as shown in Figure 2.

The model makes predictions of a latent user attribute e.g., Republican under a model assumption of sequentially arriving, independent and identically distributed observations $T = (t_1, \ldots, t_k)$[9]. The model dynamically updates posterior probability estimates $p(a(v_i) = R|t_k)$ for a given user

[5]The original dataset was collected in 2012 and has been recently released at http://icwsm.cs.mcgill.ca/. Political labels are extracted from http://www.wefollow.com as described by Pennacchiotti and Popescu (2011b).

[6]This inability to perfectly replicate prior work based on Twitter is a recognized problem throughout the community of computational social science, arising from the data policies of Twitter itself, it is not specific to this work.

[7]We use log-linear models over reasonable alternatives such as perceptron or SVM, following the practice of a wide range of previous work in related areas (Smith, 2004; Liu et al., 2005; Poon et al., 2009) including text classification in social media (Van Durme, 2012b; Yang and Eisenstein, 2013).

[8]The separate issue is that many authors simply don't tweet very often. For instance, 85.3% of all Twitter users post less than one update per day as reported at http://www.sysomos.com/insidetwitter/. Thus, their communications are scare even if we could get all of them without rate limiting from Twitter API.

[9]Given the dynamic character of online discourse it will clearly be of interest in the future to consider models that go beyond the iid assumption.

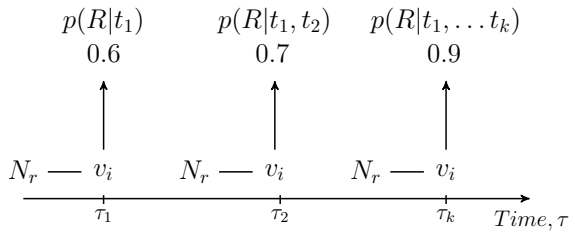


Figure 2: Stream-based classification of an attribute $a(v_i) \in \{R, D\}$ given a stream of communications $t_1, t_2, \ldots, t_k$ authored by a user v_i or user immediate neighbors from N_r social circles at time $\tau_1, \tau_2, \ldots, \tau_k$.

v_i as an additional evidence t_k is acquired, as defined in a general form below for any latent attribute $a(v_i) \in A$ given the tweets T of user v_i:

$$p(a(v_i) = x \in A \mid T) = \frac{p(T \mid a(v_i) = x) \cdot p(a(v_i) = x)}{\sum_{y \in A} p(T \mid a(v_i) = y) \cdot p(a(v_i) = y)} = \frac{\prod_k p(t_k \mid a(v_i) = x) \cdot p(a(v_i) = x)}{\sum_{y \in A} \prod_k p(t_k \mid a(v_i) = y) \cdot p(a(v_i) = y),} \quad (5)$$

where y is the number of all possible attribute values, and k is the number of tweets per user.

For example, to predict user political preference, we start with a prior $P(R) = 0.5$, and sequentially update the posterior $p(R \mid T)$ by accumulating evidence from the likelihood $p(t_k|R)$:

$$p(R \mid T) = \frac{\prod_k p(t_k|R) \cdot p(R)}{\prod_k P(t_k|R) \cdot p(R) + \prod_k P(t_k|D) \cdot p(D).} \quad (6)$$

Our goal is to maximize posterior probability estimates given a stream of communications for each user in the data over (a) time τ and (b) the number of tweets T. For that, for each user we take tweets that arrive continuously over time and apply two different streaming models:

- **User Model with Dynamic Updates:** relies exclusively on user tweets $t_1^{(v_i)}, \ldots, t_k^{(v_i)}$ following the order they arrive over time τ, where for each user v_i we dynamically update the posterior $p(R \mid t_1^{(v_i)}, \ldots, t_k^{(v_i)})$.
- **User-Neighbor Model with Dynamic Updates:** relies on both neighbor N_r communications including friend, follower, retweet, user mention and user tweets $t_1^{(v_i)}, \ldots, t_k^{(N_r)}$ following the order they arrive over time τ; here we dynamically update the posterior probability $p(R \mid t_1^{(v_i)}, \ldots, t_k^{(N_r)})$.

5 Experimental Setup

We design a set of experiments to analyze static and dynamic models for political affiliation classification defined in Sections 3 and 4.

5.1 Batch Classification Experiments

We first answer whether communications from user-local neighborhoods can help predict political preference for the user. To explore the contribution of different neighborhood types we learn static user and neighbor models on G_{cand}, G_{geo} and G_{ZLR} graphs. We also examine the ability of our static models to predict user political preferences in low-resource setting e.g., 5 tweets.

The existing models follow a standard setup when either user or neighbor tweets are available during train and test. For a static neighbor model we go beyond that, and train our the model on all data available per user, but only apply part of the data at the test time, pushing the boundaries of how little is truly required for classification. For example, we only use follower tweets for G^{test}, but we use tweets from all types of neighbors for G^{train}. Such setup will simulate different real-world prediction scenarios which have not been previously explored, to our knowledge e.g., when a user has a private profile or has not tweeted yet, and only user neighbor tweets are available.

We experiment with our static neighbor model defined in Eq.2 with the aim to:

1. evaluate neighborhood size influence, we change the number of neighbors and try $n = [1, 2, 5, 10]$ neighbor(s) per user;
2. estimate neighbor content influence, we alternate the amount of content per neighbor and try $t = [5, 10, 15, 25, 50, 100, 200]$ tweets.

We perform 10-fold cross validation[10] and run 100 random restarts for every n and t parameter combination. We compare our static neighbor and user models using the cost functions from Eq.3 and Eq.4. For all experiments we use `LibLinear` (Fan et al., 2008), integrated in the `Jerboa` toolkit (Van Durme, 2012a). Both models defined in Eq.1 and Eq.2 are learned using normalized count-based word ngram features extracted from either user or neighbor tweets.[11]

[10] For each fold we split the data into 3 parts: 70% train, 10% development and 20% test.

[11] For brevity we omit reporting results for bigram and trigram features, since unigrams showed superior performance.

5.2 Streaming Classification Experiments

We evaluate our models with dynamic Bayesian updates on a continuous stream of communications over time as shown in Figure 2. Unlike static model experiments, we are not modeling the influence of the number of neighbors or the amount of content per neighbor. Here, we order user and neighbor communication streams by real world time of posting and measure changes in posterior probabilities over time. The main purpose of these experiments is to quantitatively evaluate (1) the number of tweets and (2) the amount of real world time it takes to observe enough evidence on Twitter to make reliable predictions.

We experiment with log-linear models defined in Eq. 1 and 2 and continuously estimate the posterior probabilities $P(R \mid T)$ as defined in Eq.6. We average the posterior probability results over the users in G_{cand}, G_{geo} and G_{ZLR} graphs. We train streaming models on an attribute balanced subset of tweets for each user v_i excluding v_i's tweets (or v_i's neighbor tweets for a joint model). This setup is similar to leave-one-out classification. The classifier is learned using binary word ngram features extracted from user or user-neighbor communications. We prefer binary to normalized count-based features to overcome sparsity issues caused by making predictions on each tweet individually.

6 Static Classification Results

6.1 Modeling User Content Influence

We investigate classification decision probabilities for our static user model Φ_{v_i} by making predictions on a random set of 5 vs. 100 tweets per user. To our knowledge only limited work on personal analytics (Burger et al., 2011; Van Durme, 2012b) have performed this straight-forward comparison. For that purpose, we take a random partition containing 100 users of G_{cand} graph and perform four independent classification experiments – two runs using 5 and two runs using 100 tweets per user.

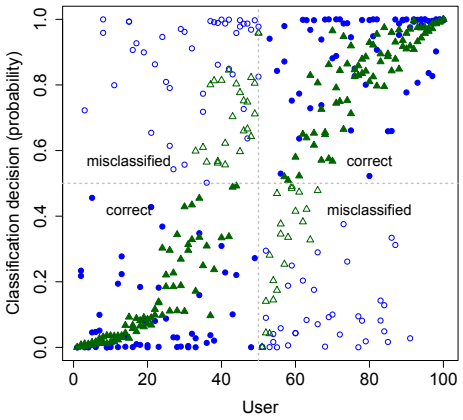
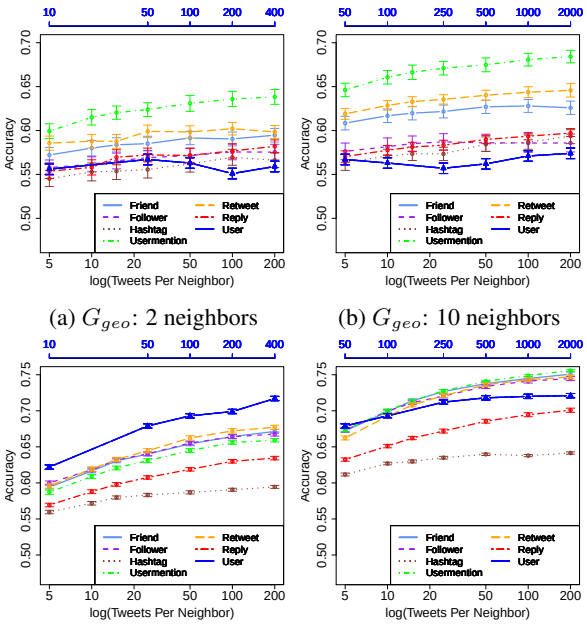


Figure 3: Classification probabilities for Φ_{v_i} estimated over 100 users in G_{cand} tested on 5 (blue) vs. 100 (green) tweets per user where Republican = 1, Democratic = 0, filled markers = correctly classified, not filled = misclassified users.



(a) G_{geo}: 2 neighbors (b) G_{geo}: 10 neighbors

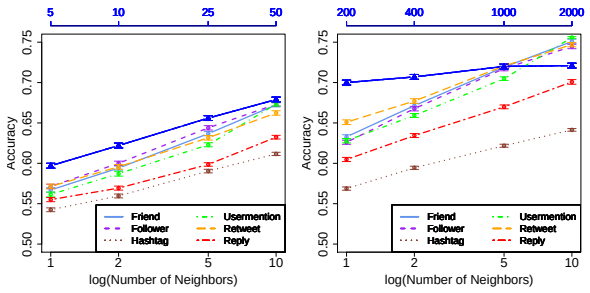
(c) G_{cand}: 2 neighbors (d) G_{cand}: 10 neighbors

Figure 4: Modeling the influence of the number of tweets per neighbor t=[5, .., 200] for G_{cand} and G_{geo} graphs.

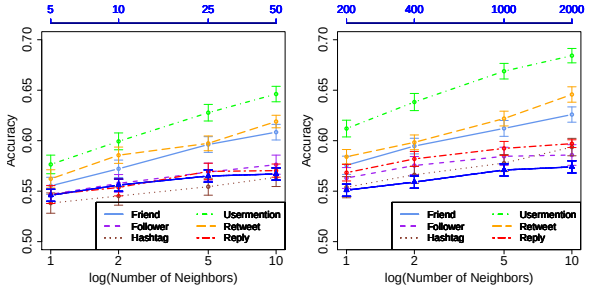
Figure 3 demonstrates that more tweets during prediction time lead to higher accuracy by showing that more users with 100 tweets are correctly classified e.g., filled green markers in the right upper quadrant are true Republicans and in the left lower quadrant are true Democrats. Moreover, a lot of users with 100 tweets are close to 0.5 decision probability which suggests that the classifier is just uncertain rather then being completely off, e.g., misclassified Republican users with 5 tweets (not filled blue markers in the right lower quadrant) are close to 0. These results follow naturally from the underlying feature representation: having more tweets per user leads to a lower variance estimate of a target multinomial distribution. The more robustly this distribution is estimated (based on having more tweets) the more confident we should be in the classifier output.

6.2 Modeling Neighbor Content Influence

Here we discuss the results for our static neighborhood model. We study the influence of the neighborhood type r and size in terms of the number of neighbors n and tweets t per neighbor.



(a) G_{cand}: 5 tweets (b) G_{cand}: 200 tweets



(c) G_{geo}: 5 tweets (d) G_{geo}: 200 tweets

Figure 5: Modeling the influence of the number of neighbors per user n=[1, .., 10] for G_{cand} and G_{geo} graphs.

In Figure 4 we present accuracy results for G_{cand} and G_{geo} graphs. Following Eq.3 and 4, we spent an equal amount of resources to obtain 100 user tweets and 10 tweets from 10 neighbors. We annotate these 'points of equal number of communications' with a line on top marked with a corresponding number of user tweets.

We show that three of six social circles – friend, retweet and user-mention yield better accuracy compared to the user model for all graphs when $t \geq 250$. Thus, for effectively classifying a given user v_i it is better to take 200 tweets each from 10 neighbors rather than 2,000 tweets from the user.

The best accuracy for G_{cand} is 0.75 for friend, follower, retweet and user-mention neighborhoods which is 0.03 higher than the user baseline; for G_{geo} is 0.67 for user-mention and 0.64 for retweet circles compared to 0.57 for the user model; for G_{ZLR} is 0.863 for retweet and 0.849 for friend circles which is 0.11 higher that the user baseline. Finally, similarly to the results for the user model given in Figure 3, increasing the number of tweets per neighbor from 5 to 200 leads to a significant gain in performance for all neighborhood types.

6.3 Modeling Neighborhood Size

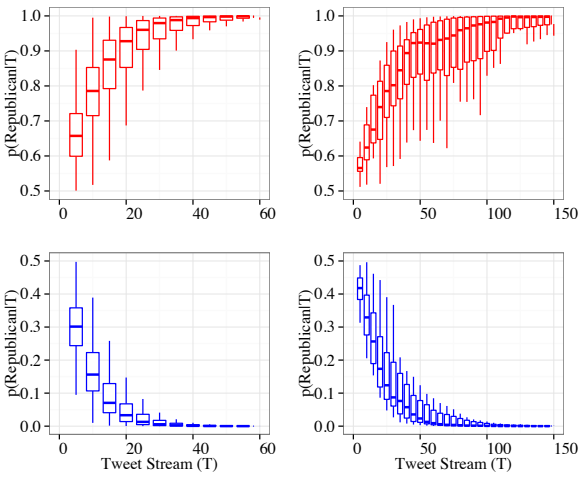
In Figure 5 we present accuracy results to show neighborhood size influence on classification performance for G_{geo} and G_{cand} graphs. Our results demonstrate that even small changes to the neighborhood size n lead to better performance which does not support the claims by Zamal et al. (2012). We demonstrate that increasing the size of the neighborhood leads to better performance across six neighborhood types. Friend, user mention and retweet neighborhoods yield the highest accuracy for all graphs. We observe that when the number of neighbors is $n = 1$, the difference in accuracy across all neighborhood types is less significant but for $n \geq 2$ it becomes more significant.

7 Streaming Classification Results

7.1 Modeling Dynamic Posterior Updates from a User Stream

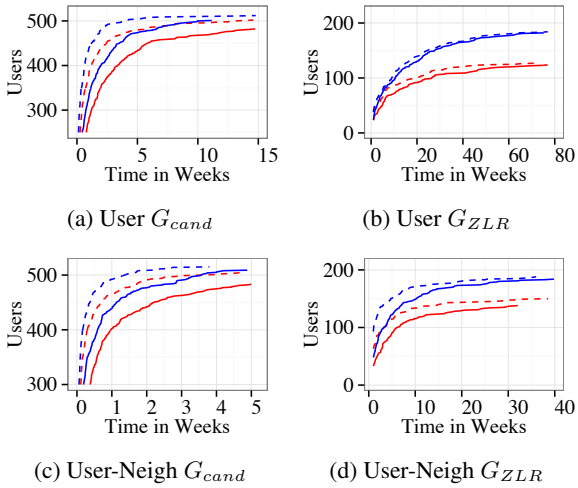
Figures 6a and 6b demonstrate dynamic user model prediction results averaged over users from G_{cand} and G_{ZLR} graphs. Each figure outlines changes in sequential average probability estimates $p_\mu(R \mid T)$ for each individual self-authored tweet t_k as defined in Eq. 6. The average probability estimates $p_\mu(R \mid T)$ are reported for every 5 tweets in a stream $T = (t_1, \ldots t_k)$ as $\frac{\sum_n P(R|t_k)}{n}$, where n is the total number of users with the same attribute R or D. We represent $p_\mu(R \mid T)$ as a box and whisker plot with the median, lower and upper quantiles to show the variance; the length of whiskers indicate lower and upper extreme values.

We find similar behavior across all three graphs. In particular, the posterior estimates converge faster when predicting Democratic than Republican users but it has been trained on an equal number of tweets per class. We observe that average posterior estimates $P_\mu(R \mid T)$ converge faster to 0



(a) User G_{cand} (b) User G_{ZLR}

Figure 6: Streaming classification results from user communications for G_{cand} and G_{ZLR} graphs averaged over every 5 tweets (red - Republican, blue - Democratic).



(a) User G_{cand} (b) User G_{ZLR}

(c) User-Neigh G_{cand} (d) User-Neigh G_{ZLR}

Figure 7: Time needed for (a) - (b) dynamic user model and (c) - (d) joint user-neighbor model to infer political preferences of Democratic (blue) and Republican (red) users at 75% (dotted line) and 95% (solid line) accuracy levels.

(Democratic) than to 1 (Republican) in Figures 6a and 6b. It suggests that language of Democrats is more expressive of their political preference than language of Republicans. For example, frequent politically influenced terms used widely by Democratic users include *faith4liberty, constitutionally, pass, vote2012, terroristic.*

The variance for average posterior estimates decreases when the number of tweets increases for all three datasets. Moreover, we detect that $P_\mu(R|T)$ estimates for users in G_{cand} converge 2-3 times faster in terms of number of tweets than for users in G_{ZLR}. The lowest convergence is detected for G_{geo} where after $t_k = 250$ tweets the average posterior estimate $P_\mu(R \mid t_k) = 0.904 \pm 0.044$ and $P_\mu(D \mid t_k) = 0.861 \pm 0.008$. It means that users in G_{cand} are more politically vocal compared to users in G_{ZLR} and G_{geo}. As a result, less active users in G_{geo} just need more than 250 tweets to converge to a true 0 or 1 class. These results are coherent with the outcomes for our static models shown in Figures 4 and 5. These findings further confirm that differences in performance are caused by various biases present in the data due to distinct sampling and annotation approaches.

Figure 7a and 7b illustrate the amount of time required for the user model to infer political preferences estimated for 1,031 users in G_{cand} and 371 users in G_{ZLR}. The amount of time needed can be evaluated for different accuracy levels e.g., 0.75 and 0.95. Thus, with 75% accuracy we classify:

- 100 ($\sim$20%) Republican users in 3.6 hours and Democratic users in 2.2 hours for G_{cand};
- 100 ($\sim$56%) R users in 20 weeks and 100 ($\sim$52%) D users in 8.9 weeks for G_{ZLR} which is 800 times longer that for G_{cand};
- 100 ($\sim$75%) R users in 12 weeks and 80 ($\sim$60%) D users in 19 weeks for G_{geo}.

Such extreme divergences in the amount of time required for classification across all graphs should be of strong interest to researchers concerned with latent attribute prediction tasks because Twitter users produce messages with extremely different frequencies. In our case, users in G_{ZLR} tweet approximately 800 times less frequently than users in G_{cand}.

7.2 Modeling Dynamic Posterior Updates from a Joint User-Neighbor Stream

We estimate dynamic posterior updates from a joint stream of user and neighbor communications in G_{geo}, G_{cand} and G_{ZLR} graphs. To make a fair comparison with a streaming user model, we start with the same user tweet $t_0(v_i)$. Then instead of waiting for the next user tweet we rely on any neighbor tweets that appear until the user produces the next tweet $t_1(v_i)$. We rely on communications from four types of neighbors such as friends, followers, retweets and user mentions.

The convergence rate for the average posterior probability estimates $P_\mu(R|T)$ depending on the number of tweets is similar to the user model results presented in Figure 6. However, for G_{geo} the variance for $P_\mu(R|T)$ is higher for Democratic users; for G_{ZLR} $P_\mu(R|T) \rightarrow 1$ for Republicans in less than 110 tweets which is $\Delta t = 40$ tweets faster than the user model; for G_{cand} the convergence for both $P_\mu(R|T) \rightarrow 1$ and $P_\mu(D|T) \rightarrow 0$ is not significantly different than the user model.

Figures 7c and 7d show the amount of time required for a joint user-neighbor model to infer political preferences estimated for users in G_{cand} and G_{ZLR}. We find that with 75% accuracy we can classify 100 users for:

- G_{cand}: Republican users in 23 minutes and Democratic users in 10 minutes;
- G_{ZLR}: R users in 3.2 weeks and D users in 1.1 weeks which is 7 times faster on average across attributes than for the user model;
- G_{geo}: R users in 1.2 weeks and D users in 3.5 weeks which is on average 6 times faster across attributes than for the user model.

Similar or better $P_\mu(R|T)$ convergence in terms of the number of tweets and, especially, in the amount of time needed for user and user-neighbor

models further confirms that neighborhood content is useful for political preference prediction. Moreover, communications from a joint stream allow to make an inference up to 7 times faster.

8 Related Work

Supervised Batch Approaches The vast majority of work on predicting latent user attributes in social media apply supervised static SVM models for discrete categorical e.g., gender and regression models for continuous attributes e.g., age with lexical bag-of-word features for classifying user gender (Garera and Yarowsky, 2009; Rao et al., 2010; Burger et al., 2011; Van Durme, 2012b), age (Rao et al., 2010; Nguyen et al., 2011; Nguyen et al., 2013) or political orientation. We present an overview of the existing models for political preference prediction in Table 1.

Bergsma et al. (2012) following up on Rao's work (2010) on adding socio-linguistic features to improve gender, ethnicity and political preference prediction show that incorporating stylistic and syntactic information to the bag-of-word features improves gender classification.

Other methods characterize Twitter users by applying limited amounts of network structure information in addition to lexical features. Conover et al. (2011) rely on identifying strong partisan clusters of Democratic and Republican users in a Twitter network based on retweet and user mention degree of connectivity, and then combine this clustering information with the follower and friend neighborhood size features. Pennacchiotti et al. (2011a; 2011b) focus on user behavior, network structure and linguistic features. Similar to our work, they assume that users from a particular class tend to reply and retweet messages of the users from the same class. We extend this assumption and study other relationship types e.g., friends, user mentions etc. Recent work by Wong et al. (2013) investigates tweeting and retweeting behavior for political learning during 2012 US Presidential election. The most similar work to ours is by Zamal et al. (2012), where the authors apply features from the tweets authored by a user's friend to infer attributes of that user. In this paper, we study different types of user social circles in addition to a friend network.

Approach	Users	Tweets	Features	Accur.
Rao et al. (2010)	1K	2M	ngrams	**0.824**
			socio-ling	0.634
			stacked	0.809
Pennacchiotti and Popescu (2011a)	10.3K	–	ling-all	0.770
			soc-all	0.863
			full	**0.889**
Conover et al. (2011)	1,000	1M	full-text	0.792
			hashtags	0.908
			clusters	**0.949**
Zamal et al. (2012)	400	400K	UserOnly	0.890
		3.85M	Nbr	0.920
		4.25M	User-Nbr[11]	**0.932**
Cohen and Ruths (2013)	397	397K	features	**0.910**
	1.8K	1.8M	from (Za-	0.840
	262	262K	mal et al.,	0.680
	196	196K	2012)	0.870
This paper (batch classification)	G_{cand}	206K	user ngrams	0.720
	1,031	2M	neighbor	**0.750**
	G_{geo}	54K	user ngrams	0.570
	270	540K	neighbor	**0.670**
	G_{ZLR}	371K	user ngrams	0.886
	371	1.5M	neighbor	**0.920**
This paper (dynamic Bayesian update classification)	G_{cand}	103K	user stream	0.995
	1,031	130K	user-neigh.	**0.999**
	G_{geo}	54K	user stream	0.843
	270	67K	user-neigh.	**0.882**
	G_{ZLR}	74K	user stream	0.892
	371	185K	user-neigh.	**0.999**

Table 1: Overview of the existing approaches for political preference classification in Twitter.

Additionally, using social media for mining political opinions (O'Connor et al., 2010a; Maynard and Funk, 2012) or understanding sociopolitical trends and voting outcomes (Tumasjan et al., 2010; Gayo-Avello, 2012; Lampos et al., 2013) is becoming a common practice. For instance, Lampos et al. (2013) propose a bilinear user-centric model for predicting voting intentions in the UK and Australia from social media data. Other works explore political blogs to predict what content will get the most comments (Yano et al., 2013) or analyze communications from Capitol Hill[12] to predict campaign contributors based on this content (Yano and Smith, 2013).

Unsupervised Batch Approaches Bergsma et al. (2013) show that large-scale clustering of user names improves gender, ethnicity and location classification on Twitter. O'Connor et al. (2010b) following the work by Eisenstein (2010) propose a Bayesian generative model to discover demographic language variations in Twitter. Rao et al. (2011) suggest a hierarchical Bayesian model which takes advantage of user name morphology for predicting user gender and ethnicity. Golbeck et al. (2010) incorporate Twitter data in a spatial model of political ideology.

Streaming Approaches Van Durme (2012b) proposed streaming models to predict user gender in Twitter. Other works suggested to process

[12]http://www.tweetcongress.org

text streams for a variety of NLP tasks e.g., real-time opinion mining and sentiment analysis in social media (Pang and Lee, 2008), named entity disambiguation (Sarmento et al., 2009), statistical machine translation (Levenberg et al., 2011), first story detection (Petrović et al., 2010), and unsupervised dependency parsing (Goyal and Daumé, 2011). Massive Online Analysis (MOA) toolkit developed by Bifet et al. (2010) is an alternative to the Jerboa package used in this work developed by Van Durme (2012a). MOA has been effectively used to detect sentiment changes in Twitter streams (Bifet et al., 2011).

9 Conclusions and Future Work

In this paper, we extensively examined state-of-the-art static approaches and proposed novel models with dynamic Bayesian updates for streaming personal analytics on Twitter. Because our streaming models rely on communications from Twitter users and content from various notions of user-local neighborhood they can be effectively applied to real-time dynamic data streams. Our results support several key findings listed below.

Neighborhood content is useful for personal analytics. Content extracted from various notions of a user-local neighborhood can be as effective or more effective for political preference classification than user self-authored content. This may be an effect of 'sparseness' of relevant user data, in that users talk about politics very sporadically compared to a random sample of their neighbors.

Substantial signal for political preference prediction is distributed in the neighborhood. Querying for more neighbors per user is more beneficial than querying for extra content from the existing neighbors e.g., 5 tweets from 10 neighbors leads to higher accuracy than 25 tweets from 2 neighbors or 50 tweets from 1 neighbor. This may be also the effect of data heterogeneity in social media compared to e.g., political debate text (Thomas et al., 2006). These findings demonstrate that a substantial signal is distributed over the neighborhood content.

Neighborhoods constructed from friend, user mention and retweet relationships are most effective. Friend, user mention and retweet neighborhoods show the best accuracy for predicting political preferences of Twitter users. We think that friend relationships are more effective than e.g., follower relationships because it is very likely that users share common interests and preferences with their friends, e.g. Facebook friends can even be used to predict a user's credit score.[13] User mentions and retweets are two primary ways of interaction on Twitter. They both allow to share information e.g., political news, events with others and to be involved in direct communication e.g., live political discussions, political groups.

Streaming models are more effective than batch models for personal analytics. The predictions made using dynamic models with Bayesian updates over user and joint user-neighbor communication streams demonstrate higher performance with lower resources spent compared to the batch models. Depending on user political involvement, expressiveness and activeness, the perfect prediction (approaching 100% accuracy) can be made using only 100 - 500 tweets per user.

Generalization of the classifiers for political preference prediction. This work raises a very important but under-explored problem of the generalization of classifiers for personal analytics in social media, also recently discussed by Cohen and Ruth (2013). For instance, the existing models developed for political preference prediction are all trained on Twitter data but report significantly different results even for the same baseline models trained using bag-of-word lexical features as shown in Table 1. In this work we experiment with three different datasets. Our results for both static and dynamic models show that the accuracy indeed depends on the way the data was constructed. Therefore, publicly available datasets need to be released for a meaningful comparison of the approaches for personal analytics in social media.

In future work, we plan to incorporate iterative model updates from newly classified communications similar to online perceptron-style updates. In addition, we aim to experiment with neighborhood-specific classifiers applied towards the tweets from neighborhood-specific streams e.g., friend classifier used for friend tweets, retweet classifier applied to retweet tweets etc.

Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful comments.

[13] http://money.cnn.com/2013/08/26/technology/social/facebook-credit-score/

References

Shane Bergsma, Matt Post, and David Yarowsky. 2012. Stylometric analysis of scientific articles. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 327–337.

Shane Bergsma, Mark Dredze, Benjamin Van Durme, Theresa Wilson, and David Yarowsky. 2013. Broadly improving user classification via communication-based name and location clustering on Twitter. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1010–1019.

Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Philipp Kranen, Hardy Kremer, Timm Jansen, and Thomas Seidl. 2010. MOA: Massive online analysis, a framework for stream classification and clustering. *Journal of Machine Learning Research*, 11:44–50.

Albert Bifet, Geoffrey Holmes, Bernhard Pfahringer, and Ricard Gavaldà. 2011. Detecting sentiment change in Twitter streaming data. *Journal of Machine Learning Research*, 17:5–11.

John D. Burger, John Henderson, George Kim, and Guido Zarrella. 2011. Discriminating gender on Twitter. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1301–1309.

Raviv Cohen and Derek Ruths. 2013. Classifying Political Orientation on Twitter: It's Not Easy! In *Proceedings of the International AAAI Conference on Weblogs and Social Media (ICWSM)*, pages 91–99.

Michael D. Conover, Bruno Gonçalves, Jacob Ratkiewicz, Alessandro Flammini, and Filippo Menczer. 2011. Predicting the political alignment of Twitter users. In *Proceedings of Social Computing*, pages 192–199.

Jacob Eisenstein, Brendan O'Connor, Noah A. Smith, and Eric P. Xing. 2010. A latent variable model for geographic lexical variation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1277–1287.

Rong En Fan, Kai Wei Chang, Cho Jui Hsieh, Xiang Rui Wang, and Chih Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Katja Filippova. 2012. User demographics and language in an implicit social network. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1478–1488.

Nikesh Garera and David Yarowsky. 2009. Modeling latent biographic attributes in conversational genres. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 710–718.

Daniel Gayo-Avello. 2012. No, you cannot predict elections with Twitter. *Internet Computing, IEEE*, 16(6):91–94.

Jennifer Golbeck, Justin M. Grimes, and Anthony Rogers. 2010. Twitter use by the u.s. congress. *Journal of the American Society for Information Science and Technology*, 61(8):1612–1621.

Amit Goyal and Hal Daumé, III. 2011. Approximate scalable bounded space sketch for large data NLP. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 250–261.

Vasileios Lampos, Daniel Preotiuc-Pietro, and Trevor Cohn. 2013. A user-centric model of voting intention from social media. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 993–1003.

Abby Levenberg, Miles Osborne, and David Matthews. 2011. Multiple-stream language models for statistical machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation (WMT)*, pages 177–186.

Yang Liu, Qun Liu, and Shouxun Lin. 2005. Log-linear models for word alignment. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL)*, pages 459–466.

Ingrid Lunden. 2012. Analyst: Twitter passed 500M users in june 2012, 140m of them in US; Jakarta 'biggest tweeting' city. http://techcrunch.com/2012/07/30/analyst-twitter-passed-500m-users-in-june-2012-140m-of-them-in-us-jakarta-biggest-tweeting-city/.

Diana Maynard and Adam Funk. 2012. Automatic detection of political opinions in tweets. In *Proceedings of the 8th International Conference on The Semantic Web (ESWC)*, pages 88–99.

Felix Ming Fai Wong, Chee Wei Tan, Soumya Sen, and Mung Chiang. 2013. Quantifying political leaning from tweets and retweets. In *Proceedings of the International AAAI Conference on Weblogs and Social Media (ICWSM)*.

Dong Nguyen, Noah A. Smith, and Carolyn P. Rosé. 2011. Author age prediction from text using linear regression. In *Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH)*, pages 115–123.

Dong Nguyen, Rilana Gravel, Dolf Trieschnigg, and Theo Meder. 2013. "How old do you think I am?" A study of language and age in Twitter. In *Proceedings of the AAAI Conference on Weblogs and Social Media (ICWSM)*, pages 439–448.

Brendan O'Connor, Ramnath Balasubramanyan, Bryan R. Routledge, and Noah A. Smith. 2010a. From tweets to polls: Linking text sentiment to public opinion time series. In *Proceedings of the International AAAI Conference on Weblogs and Social Media (ICWSM)*, pages 122–129.

Brendan O'Connor, Jacob Eisenstein, Eric P. Xing, and Noah A. Smith. 2010b. A mixture model of demographic lexical variation. In *Proceedings of the NIPS Workshop on Machine Learning and Social Computing*.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations of Trends in Information Retrieval*, 2(1-2):1–135, January.

Marco Pennacchiotti and Ana-Maria Popescu. 2011a. Democrats, republicans and starbucks afficionados: user classification in twitter. In *Proceedings of the 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 430–438.

Marco Pennacchiotti and Ana Maria Popescu. 2011b. A machine learning approach to Twitter user classification. In *Proceedings of the International AAAI Conference on Weblogs and Social Media (ICWSM)*, pages 281–288.

Saša Petrović, Miles Osborne, and Victor Lavrenko. 2010. Streaming first story detection with application to Twitter. In *Proceedings of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Hoifung Poon, Colin Cherry, and Kristina Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 209–217.

Delip Rao, David Yarowsky, Abhishek Shreevats, and Manaswi Gupta. 2010. Classifying latent user attributes in Twitter. In *Proceedings of the 2nd International Workshop on Search and Mining User-generated Contents (SMUC)*, pages 37–44.

Delip Rao, Michael Paul, Clay Fink, David Yarowsky, Timothy Oates, and Glen Coppersmith. 2011. Hierarchical Bayesian models for latent attribute detection in social media. In *Proceedings of the International AAAI Conference on Weblogs and Social Media (ICWSM)*.

Luís Sarmento, Alexander Kehlenbeck, Eugénio Oliveira, and Lyle Ungar. 2009. An approach to web-scale named-entity disambiguation. In *Proceedings of the 6th International Conference on Machine Learning and Data Mining in Pattern Recognition (MLDM)*, pages 689–703.

Noah A. Smith. 2004. Log-linear models.

Craig Smith. 2013. May 2013 by the numbers: 16 amazing Twitter stats. http://expandedramblings.com/index.php/march-2013-by-the-numbers-a-few-amazing-twitter-stats/.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: determining support or opposition from congressional floor-debate transcripts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 327–335.

A. Tumasjan, T. O. Sprenger, P. G. Sandner, and I. M. Welpe. 2010. Predicting elections with Twitter: What 140 characters reveal about political sentiment. In *Proceedings of the International AAAI Conference on Weblogs and Social Media*, pages 178–185.

Benjamin Van Durme. 2012a. Jerboa: A toolkit for randomized and streaming algorithms. Technical report, Human Language Technology Center of Excellence.

Benjamin Van Durme. 2012b. Streaming analysis of discourse participants. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 48–58.

Yi Yang and Jacob Eisenstein. 2013. A log-linear model for unsupervised text normalization. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, pages 61–72.

Tao Yano and Noah A. Smith. 2013. What's worthy of comment? content and comment volume in political blogs. In *International AAAI Conference on Weblogs and Social Media (ICWSM)*.

Tao Yano, Dani Yogatama, and Noah A. Smith. 2013. A penny for your tweets: Campaign contributions and capitol hill microblogs. In *Proceedings of the International AAAI Conference on Weblogs and Social Media (ICWSM)*.

Faiyaz Al Zamal, Wendy Liu, and Derek Ruths. 2012. Homophily and latent attribute inference: Inferring latent attributes of Twitter users from neighbors. In *Proceedings of the International AAAI Conference on Weblogs and Social Media*, pages 387–390.

Steps to Excellence: Simple Inference with Refined Scoring of Dependency Trees

Yuan Zhang, Tao Lei, Regina Barzilay, Tommi Jaakkola
Massachusetts Institute of Technology
{yuanzh, taolei, regina, tommi}@csail.mit.edu

Amir Globerson
The Hebrew University
gamir@cs.huji.ac.il

Abstract

Much of the recent work on dependency parsing has been focused on solving inherent combinatorial problems associated with rich scoring functions. In contrast, we demonstrate that highly expressive scoring functions can be used with substantially simpler inference procedures. Specifically, we introduce a sampling-based parser that can easily handle arbitrary global features. Inspired by SampleRank, we learn to take guided stochastic steps towards a high scoring parse. We introduce two samplers for traversing the space of trees, Gibbs and Metropolis-Hastings with Random Walk. The model outperforms state-of-the-art results when evaluated on 14 languages of non-projective CoNLL datasets. Our sampling-based approach naturally extends to joint prediction scenarios, such as joint parsing and POS correction. The resulting method outperforms the best reported results on the CATiB dataset, approaching performance of parsing with gold tags.[1]

1 Introduction

Dependency parsing is commonly cast as a maximization problem over a parameterized scoring function. In this view, the use of more expressive scoring functions leads to more challenging combinatorial problems of finding the maximizing parse. Much of the recent work on parsing has been focused on improving methods for solving the combinatorial maximization inference problems. Indeed, state-of-the-art results have been obtained by adapting powerful tools from optimization (Martins et al., 2013; Martins et al., 2011; Rush and Petrov, 2012). We depart from this view and instead focus on using highly expressive scoring functions with substantially simpler inference procedures. The key ingredient in our approach is how learning is coupled with inference. Our combination outperforms the state-of-the-art parsers and remains comparable even if we adopt their scoring functions.

Rich scoring functions have been used for some time. They first appeared in the context of reranking (Collins, 2000), where a simple parser is used to generate a candidate list which is then reranked according to the scoring function. Because the number of alternatives is small, the scoring function could in principle involve arbitrary (global) features of parse trees. The power of this methodology is nevertheless limited by the initial set of alternatives from the simpler parser. Indeed, the set may already omit the gold parse. We dispense with the notion of a candidate set and seek to exploit the scoring function more directly.

In this paper, we introduce a sampling-based parser that places few or no constraints on the scoring function. Starting with an initial candidate tree, our inference procedure climbs the scoring function in small (cheap) stochastic steps towards a high scoring parse. The proposal distribution over the moves is derived from the scoring function itself. Because the steps are small, the complexity of the scoring function has limited impact on the computational cost of the procedure. We explore two alternative proposal distributions. Our first strategy is akin to Gibbs sampling and samples a new head for each word in the sentence, modifying one arc at a time. The second strategy relies on a provably correct sampler for first-order scores (Wilson, 1996), and uses it within a Metropolis-Hastings algorithm for general scoring functions. It turns out that the latter optimizes the

[1] The source code for the work is available at http://groups.csail.mit.edu/rbg/code/global/acl2014.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 197–207,
Baltimore, Maryland, USA, June 23-25 2014.

score more efficiently than the former.

Because the inference procedure is so simple, it is important that the parameters of the scoring function are chosen in a manner that facilitates how we climb the scoring function in small steps. One way to achieve this is to make sure that improvements in the scoring functions are correlated with improvements in the quality of the parse. This approach was suggested in the SampleRank framework (Wick et al., 2011) for training structured prediction models. This method was originally developed for a sequence labeling task with local features, and was shown to be more effective than state-of-the-art alternatives. Here we apply SampleRank to parsing, applying several modifications such as the proposal distributions mentioned earlier.

The benefits of sampling-based learning go beyond stand-alone parsing. For instance, we can use the framework to correct preprocessing mistakes in features such as part-of-speech (POS) tags. In this case, we combine the scoring function for trees with a stand-alone tagging model. When proposing a small move, i.e., sampling a head of the word, we can also jointly sample its POS tag from a set of alternatives provided by the tagger. As a result, the selected tag is influenced by a broad syntactic context above and beyond the initial tagging model and is directly optimized to improve parsing performance. Our joint parsing-tagging model provides an alternative to the widely-adopted pipeline setup.

We evaluate our method on benchmark multilingual dependency corpora. Our method outperforms the Turbo parser across 14 languages on average by 0.5%. On four languages, we top the best published results. Our method provides a more effective mechanism for handling global features than reranking, outperforming it by 1.3%. In terms of joint parsing and tagging on the CATiB dataset, we nearly bridge (88.38%) the gap between independently predicted (86.95%) and gold tags (88.45%). This is better than the best published results in the 2013 SPMRL shared task (Seddah et al., 2013), including parser ensembles.

2 Related Work

Earlier works on dependency parsing focused on inference with tractable scoring functions. For instance, a scoring function that operates over each single dependency can be optimized using the maximum spanning tree algorithm (McDonald et al., 2005). It was soon realized that using higher order features could be beneficial, even at the cost of using approximate inference and sacrificing optimality. The first successful approach in this arena was reranking (Collins, 2000; Charniak and Johnson, 2005) on constituency parsing. Reranking can be combined with an arbitrary scoring function, and thus can easily incorporate global features over the entire parse tree. Its main disadvantage is that the output parse can only be one of the few parses passed to the reranker.

Recent work has focused on more powerful inference mechanisms that consider the full search space (Zhang and McDonald, 2012; Rush and Petrov, 2012; Koo et al., 2010; Huang, 2008). For instance, Nakagawa (2007) deals with tractability issues by using sampling to approximate marginals. Another example is the dual decomposition (DD) framework (Koo et al., 2010; Martins et al., 2011). The idea in DD is to decompose the hard maximization problem into smaller parts that can be efficiently maximized and enforce agreement among these via Lagrange multipliers. The method is essentially equivalent to linear programming relaxation approaches (Martins et al., 2009; Sontag et al., 2011), and also similar in spirit to ILP approaches (Punyakanok et al., 2004).

A natural approach to approximate global inference is via search. For instance, a transition-based parsing system (Zhang and Nivre, 2011) incrementally constructs a parsing structure using greedy beam-search. Other approaches operate over full trees and generate a sequence of candidates that successively increase the score (Daumé III et al., 2009; Li et al., 2013; Wick et al., 2011). Our work builds on one such approach — SampleRank (Wick et al., 2011), a sampling-based learning algorithm. In SampleRank, the parameters are adjusted so as to guide the sequence of candidates closer to the target structure along the search path. The method has been successfully used in sequence labeling and machine translation (Haddow et al., 2011). In this paper, we demonstrate how to adapt the method for parsing with rich scoring functions.

3 Sampling-Based Dependency Parsing with Global Features

In this section, we introduce our novel sampling-based dependency parser which can incorporate

arbitrary global features. We begin with the notation before addressing the decoding and learning algorithms. Finally, we extend our model to a joint parsing and POS correction task.

3.1 Notations

We denote sentences by x and the corresponding dependency trees by $y \in \mathcal{Y}(x)$. Here $\mathcal{Y}(x)$ is the set of valid (projective or non-projective) dependency trees for sentence x. We use x_j to refer to the jth word of sentence x, and h_j to the head word of x_j. A training set of size N is given as a set of pairs $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$ where $y^{(i)}$ is the ground truth parse for sentence $x^{(i)}$.

We parameterize the scoring function $s(x, y)$ as

$$s(x, y) = \theta \cdot f(x, y) \tag{1}$$

where $f(x, y)$ is the feature vector associated with tree y for sentence x. We do not make any assumptions about how the feature function decomposes. In contrast, most state-of-the-art parsers operate under the assumption that the feature function decomposes into a sum of simpler terms. For example, in the second-order MST parser (McDonald and Pereira, 2006), all the feature terms involve arcs or consecutive siblings. Similarly, parsers based on dual decomposition (Martins et al., 2011; Koo et al., 2010) assume that $s(x, y)$ decomposes into a sum of terms where each term can be maximized over y efficiently.

3.2 Decoding

The decoding problem consists of finding a valid dependency tree $y \in \mathcal{Y}(x)$ that maximizes the score $s(x, y) = \theta \cdot f(x, y)$ with parameters θ. For scoring functions that extend beyond first-order arc preferences, finding the maximizing non-projective tree is known to be NP-hard (McDonald and Pereira, 2006). We find a high scoring tree through sampling, and (later) learn the parameters θ so as to further guide this process.

Our sampler generates a sequence of dependency structures so as to approximate independent samples from

$$p(y|x, T, \theta) \propto \exp\left(s(x, y)/T\right) \tag{2}$$

The temperature parameter T controls how concentrated the samples are around the maximum of $s(x, y)$ (e.g., see Geman and Geman (1984)). Sampling from target distribution p is typically as hard as (or harder than) that maximizing $s(x, y)$.

Inputs: θ, x, T_0 (initial temperature), c (temperature update rate), proposal distribution q.
Outputs: y^*
$T \leftarrow T_0$
Set y^0 to some random tree
$y^* \leftarrow y^0$
repeat
  $y' \leftarrow q(\cdot|x, y^t, T, \theta)$
  if $s(x, y') > s(x, y^*)$ **then**
    $y^* \leftarrow y'$
  $\alpha = \min\left[1, \frac{p(y')q(y^t|y')}{p(y^t)q(y'|y^t)}\right]$
  Sample Bernouli variable Z with $P[Z = 1] = \alpha$.
  if $Z = 0$ **then**
    $y^{t+1} \leftarrow y^t$
  else
    $y^{t+1} \leftarrow y'$
  $t \leftarrow t + 1$
  $T \leftarrow c \cdot T$
until convergence
return y^*

Figure 1: Sampling-based algorithm for decoding (i.e., approximately maximizing $s(x, y)$).

We follow here a Metropolis-Hastings sampling algorithm (e.g., see Andrieu et al. (2003)) and explore different alternative proposal distributions $q(y'|x, y, \theta, T)$. The distribution q governs the small steps that are taken in generating a sequence of structures. The target distribution p folds into the procedure by defining the probability that we will accept the proposed move. The general structure of our sampling algorithm is given in Figure 1.

3.2.1 Gibbs Sampling

Perhaps the most natural choice of the proposal distribution q is a conditional distribution from p. This is feasible if we restrict the proposed moves to only small changes in the current tree. In our case, we choose a word j randomly, and then sample its head h_j according to p with the constraint that we obtain a valid tree (when projective trees are sought, this constraint is also incorporated). For this choice of q, the probability of accepting the new tree (α in Figure 1) is identically one. Thus new moves are always accepted.

3.2.2 Exact First-Order Sampling

One shortcoming of the Gibbs sampler is that it only changes one variable (arc) at a time. This usually leads to slow mixing, requiring more samples to get close to the parse with maximum score. Ideally, we would change multiple heads in the parse tree simultaneously, and sample those choices from the corresponding conditional distribution of p. While in general this is increasingly difficult with more heads, it is indeed tractable if

Inputs: x, y^t, θ, K (number of heads to change).
Outputs: y'
for $i = 1$ **to** $|x|$ **do**
  $inTree[i] \leftarrow false$
  $ChangeNode[i] \leftarrow false$
Set $ChangeNode$ to true for K random nodes.
$head[0] \leftarrow -1$
for $i = 1$ **to** $|x|$ **do**
  $u \leftarrow i$
  while not $inTree[u]$ **do**
    if $ChangeNode[u]$ **then**
      $head[u] \leftarrow$ randomHead(u, θ)
    else
      $head[u] \leftarrow y^t(u)$
    $u \leftarrow head[u]$
    if LoopExist$(head)$ **then**
      EraseLoop$(head)$
  $u \leftarrow i$
  while not $inTree[u]$ **do**
    $inTree[u] \leftarrow true$
    $u \leftarrow head[u]$
return Construct tree y' from the head array.

Figure 2: A proposal distribution $q(y'|y^t)$ based on the random walk sampler of Wilson (1996). The function randomHead samples a new head for node u according to the first-order weights given by θ.

the model corresponds to a first-order parser. One such sampling algorithm is the random walk sampler of Wilson (1996). It can be used to obtain i.i.d. samples from distributions of the form:

$$p(y) \propto \prod_{i \to j \in y} w_{ij}, \qquad (3)$$

where y corresponds to a tree with a spcified root and w_{ij} is the exponential of the first-order score. y is always a valid parse tree if we allow multiple children of the root and do not impose projective constraint. The algorithm in Wilson (1996) iterates over all the nodes, and for each node performs a random walk according to the weights w_{ij} until the walk creates a loop or hits a tree. In the first case the algorithm erases the loop and continues the walk. If the walk hits the current tree, the walk path is added to form a new tree with more nodes. This is repeated until all the nodes are included in the tree. It can be shown that this procedure generates i.i.d. trees from $p(y)$.

Since our features do not by design correspond to a first-order parser, we cannot use the Wilson algorithm as it is. Instead we use it as the proposal function and sample a subset of the dependencies from the first-order distribution of our model, while fixing the others. In each step we uniformly sample K nodes to update and sample their new

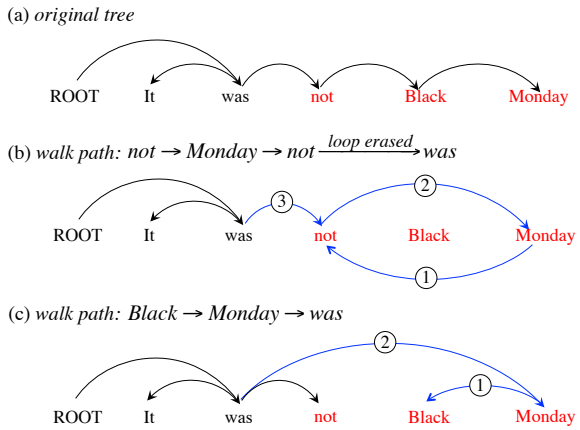


Figure 3: An illustration of random walk sampler. The index on each edge indicates its order on each walk path. The heads of the red words are sampled while others are fixed. The blue edges represent the current walk path and the black ones are already in the tree. Note that the walk direction is opposite to the dependency direction. (a) shows the original tree before sampling; (b) and (c) show the walk path and how the tree is generated in two steps. The loop *not*→ *Monday* → *not* in (b) is erased.

heads using the Wilson algorithm (in the experiments we use $K = 4$). Note that blocked Gibbs sampling would be exponential in K, and is thus very slow already at $K = 4$. The procedure is described in Figure 2 with a graphic illustration in Figure 3.

3.3 Training

In this section, we describe how to learn the adjustable parameters θ in the scoring function. The parameters are learned in an on-line fashion by successively imposing soft constraints between pairs of dependency structures. We introduce both margin constraints and constraints pertaining to successive samples generated along the search path. We demonstrate later that both types of constraints are essential.

We begin with the standard margin constraints. An ideal scoring function would always rank the gold parse higher than any alternative. Moreover, alternatives that are far from the gold parse should score even lower. As a result, we require that

$$s(x^{(i)}, y^{(i)}) - s(x^{(i)}, y) \geq \Delta(y^{(i)}, y) \quad \forall y \qquad (4)$$

where $\Delta(y^{(i)}, y)$ is the number of head mistakes in y relative to the gold parse $y^{(i)}$. We adopt here a shorthand $Err(y) = \Delta(y^{(i)}, y)$, where the de-

pendence on $y^{(i)}$ is implied from context. Note that Equation 4 contains exponentially many constraints and cannot be enforced jointly for general scoring functions. However, our sampling procedure generates a small number of structures along the search path. We enforce only constraints corresponding to those samples.

The second type of constraints are enforced between successive samples along the search path. To illustrate the idea, consider a parse y that differs from $y^{(i)}$ in only one arc, and a parse y' that differs from $y^{(i)}$ in ten arcs. We cannot necessarily assume that $s(x, y)$ is greater than $s(x, y')$ without additional encouragement. Thus, we can complement the constraints in Equation 4 with additional pairwise constraints (Wick et al., 2011):

$$s(x^{(i)}, y) - s(x^{(i)}, y') \geq Err(y') - Err(y) \quad (5)$$

where similarly to Equation 4, the difference in scores scales with the differences in errors with respect to the target $y^{(i)}$. We only enforce the above constraints for y, y' that are consecutive samples in the course of the sampling process. These constraints serve to guide the sampling process derived from the scoring function towards the gold parse.

We learn the parameters θ in an on-line fashion to satisfy the above constraints. This is done via the MIRA algorithm (Crammer and Singer, 2003). Specifically, if the current parameters are θ_t, and we enforce constraint Equation 5 for a particular pair y, y', then we will find θ_{t+1} that minimizes

$$\begin{array}{ll} \min & ||\theta - \theta_t||^2 + C\xi \\ \text{s.t.} & \theta \cdot (f(x, y) - f(x, y')) \geq Err(y') - Err(y) - \xi \end{array} \quad (6)$$

The updates can be calculated in closed form. Figure 4 summarizes the learning algorithm. We repeatedly generate parses based on the current parameters θ_t for each sentence $x^{(i)}$, and use successive samples to enforce constraints in Equation 4 and Equation 5 one at a time.

3.4 Joint Parsing and POS Correction

It is easy to extend our sampling-based parsing framework to joint prediction of parsing and other labels. Specifically, when sampling the new heads, we can also sample the values of other variables at the same time. For instance, we can sample the POS tag, the dependency relation or morphology information. In this work, we investigate a joint POS correction scenario in which only the predicted POS tags are provided in the testing phase, while both gold and predicted tags are available for the training set.

Inputs: $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$.
Outputs: Learned parameters θ.
$\theta_0 \leftarrow \mathbf{0}$
for $e = 1$ **to** #epochs **do**
  for $i = 1$ **to** N **do**
    $y' \leftarrow q(\cdot|x^{(i)}, y_i^{t_i}, \theta_t)$
    $y^+ = \arg\min_{y \in \left\{y_i^{t_i}, y'\right\}} Err(y)$
    $y^- = \arg\max_{y \in \left\{y_i^{t_i}, y'\right\}} Err(y)$
    $y_i^{t_i+1} \leftarrow$ acceptOrReject($y', y_i^{t_i}, \theta_t$)
    $t_i \leftarrow t_i + 1$
    $\nabla f = f(x^{(i)}, y^+) - f(x^{(i)}, y^-)$
    $\Delta Err = Err(y^+) - Err(y^-)$
    if $\Delta Err \neq 0$ and $\theta_t \cdot \nabla f < \Delta Err$ **then**
      $\theta_{t+1} \leftarrow$ updateMIRA($\nabla f, \Delta Err, \theta_t$)
      $t \leftarrow t + 1$
    $\nabla f_g = f(x^{(i)}, y^{(i)}) - f(x^{(i)}, y_i^{t_i})$
    if $\theta_t \cdot \nabla f_g < Err(y_i^{t_i})$ **then**
      $\theta_{t+1} \leftarrow$ updateMIRA($\nabla f_g, Err(y_i^{t_i}), \theta_t$)
      $t \leftarrow t + 1$
return Average of $\theta_0, \ldots, \theta_t$ parameters.

Figure 4: SampleRank algorithm for learning. The rejection strategy is as in Figure 1. $y_i^{t_i}$ is the t_ith tree sample of $x^{(i)}$. The first MIRA update (see Equation 6) enforces a ranking constraint between two sampled parses. The second MIRA update enforces constraints between a sampled parse and the gold parse. In practice several samples are drawn for each sentence in each epoch.

We extend our model such that it jointly learns how to predict a parse tree and also correct the predicted POS tags for a better parsing performance. We generate the POS candidate list for each word based on the confusion matrix on the training set. Let $c(t_g, t_p)$ be the count when the gold tag is t_g and the predicted one is t_p. For each word w, we first prune out its POS candidates by using the vocabulary from the training set. We don't prune anything if w is unseen. Assuming that the predicted tag for w is t_p, we further remove those tags t if their counts are smaller than some threshold $c(t, t_p) < \alpha \cdot c(t_p, t_p)$[2].

After generating the candidate lists for each word, the rest of the extension is rather straightforward. For each sampling, let $\mathcal{H}$ be the set of candidate heads and $\mathcal{T}$ be the set of candidate POS tags. The Gibbs sampler will generate a new sample from the space $\mathcal{H} \times \mathcal{T}$. The other parts of the algorithm remain the same.

[2]In our work we choose $\alpha = 0.003$, which gives a 98.9% oracle POS tagging accuracy on the CATiB development set.

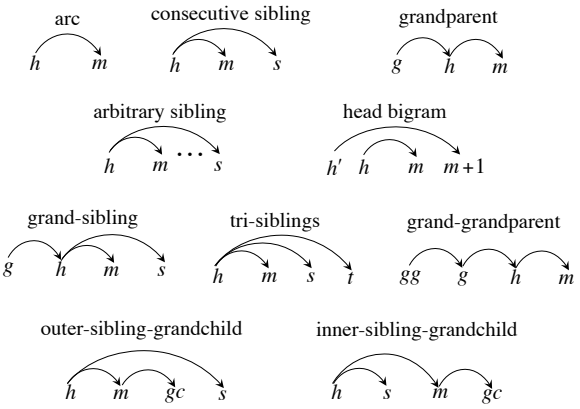


Figure 5: First- to third-order features.

4 Features

First- to Third-Order Features The feature templates of first- to third-order features are mainly drawn from previous work on graph-based parsing (McDonald and Pereira, 2006), transition-based parsing (Nivre et al., 2006) and dual decomposition-based parsing (Martins et al., 2011). As shown in Figure 5, the *arc* is the basic structure for first-order features. We also define features based on consecutive sibling, grandparent, arbitrary sibling, head bigram, grand-sibling and tri-siblings, which are also used in the Turbo parser (Martins et al., 2013). In addition to these first- to third-order structures, we also consider grand-grandparent and sibling-grandchild structures. There are two types of sibling-grandchild structures: (1) inner-sibling when the sibling is between the head and the modifier and (2) outer-sibling for the other cases.

Global Features We used feature shown promising in prior reranking work Charniak and Johnson (2005), Collins (2000) and Huang (2008).

- **Right Branch** This feature enables the model to prefer right or left-branching trees. It counts the number of words on the path from the root node to the right-most non-punctuation word, normalized by the length of the sentence.
- **Coordination** In a coordinate structure, the two adjacent conjuncts usually agree with each other on POS tags and their span lengths. For instance, in *cats and dogs*, the conjuncts are both short noun phrases. Therefore, we add different features to capture POS tag and span length consistency in a coordinate structure.
- **PP Attachment** We add features of lexical tuples involving the head, the argument and the preposition of prepositional phrases. Generally, this feature can be defined based on an instance of grandparent structure. However, we also handle the case of coordination. In this case, the arguments should be the conjuncts rather than the coordinator. Figure 6 shows an example.

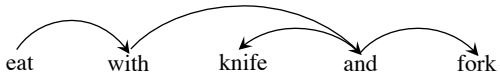


Figure 6: An example of PP attachment with coordination. The arguments should be *knife* and *fork*, not *and*.

- **Span Length** This feature captures the distribution of the binned span length of each POS tag. It also includes flags of whether the span reaches the end of the sentence and whether the span is followed by the punctuation.
- **Neighbors** The POS tags of the neighboring words to the left and right of each span, together with the binned span length and the POS tag at the span root.
- **Valency** We consider valency features for each POS tag. Specifically, we add two types of valency information: (1) the binned number of non-punctuation modifiers and (2) the concatenated POS string of all those modifiers.
- **Non-projective Arcs** A flag indicating if a dependency is projective or not (i.e. if it spans a word that does not descend from its head) (Martins et al., 2011). This flag is also combined with the POS tags or the lexical words of the head and the modifier.

POS Tag Features In the joint POS correction scenario, we also add additional features specifically for POS prediction. The feature templates are inspired by previous feature-rich POS tagging work (Toutanova et al., 2003). However, we are free to add higher order features because we do not rely on dynamic programming decoding. In our work we use feature templates up to 5-gram. Table 1 summarizes all POS tag feature templates.

5 Experimental Setup

Datasets We evaluate our model on standard benchmark corpora — CoNLL 2006 and CoNLL 2008 (Buchholz and Marsi, 2006; Surdeanu et al., 2008) — which include dependency treebanks for 14 different languages. Most of these data sets

1-gram	$\langle t_i \rangle, \langle t_i, w_{i-2} \rangle, \langle t_i, w_{i-1} \rangle, \langle t_i, w_i \rangle, \langle t_i, w_{i+1} \rangle, \langle t_i, w_{i+2} \rangle$
2-gram	$\langle t_{i-1}, t_i \rangle, \langle t_{i-2}, t_i \rangle, \langle t_{i-1}, t_i, w_{i-1} \rangle, \langle t_{i-1}, t_i, w_i \rangle$
3-gram	$\langle t_{i-1}, t_i, t_{i+1} \rangle, \langle t_{i-2}, t_i, t_{i+1}, \rangle, \langle t_{i-1}, t_i, t_{i+2} \rangle, \langle t_{i-2}, t_i, t_{i+2} \rangle$
4-gram	$\langle t_{i-2}, t_{i-1}, t_i, t_{i+1} \rangle, \langle t_{i-2}, t_{i-1}, t_i, t_{i+2} \rangle, \langle t_{i-2}, t_i, t_{i+1}, t_{i+2} \rangle$
5-gram	$\langle t_{i-2}, t_{i-1}, t_i, t_{i+1}, t_{i+2} \rangle$

Table 1: POS tag feature templates. t_i and w_i denotes the POS tag and the word at the current position. t_{i-x} and t_{i+x} denote the left and right context tags, and similarly for words.

contain non-projective dependency trees. We use all sentences in CoNLL datasets during training and testing. We also use the Columbia Arabic Treebank (CATiB) (Marton et al., 2013). CATiB mostly includes projective trees. The trees are annotated with both gold and predicted versions of POS tags and morphology information. Following Marton et al. (2013), for this dataset we use 12 core POS tags, word lemmas, determiner features, rationality features and functional genders and numbers.

Some CATiB sentences exceed 200 tokens. For efficiency, we limit the sentence length to 70 tokens in training and development sets. However, we do not impose this constraint during testing. We handle long sentences during testing by applying a simple split-merge strategy. We split the sentence based on the ending punctuation, predict the parse tree for each segment and group the roots of resulting trees into a single node.

Evaluation Measures Following standard practice, we use Unlabeled Attachment Score (UAS) as the evaluation metric in all our experiments. We report UAS excluding punctuation on CoNLL datasets, following Martins et al. (2013). For the CATiB dataset, we report UAS including punctuation in order to be consistent with the published results in the 2013 SPMRL shared task (Seddah et al., 2013).

Baselines We compare our model with the Turbo parser and the MST parser. For the Turbo parser, we directly compare with the recent published results in (Martins et al., 2013). For the MST parser, we train a second-order non-projective model using the most recent version of the code[3].

We also compare our model against a discriminative reranker. The reranker operates over the top-50 list obtained from the MST parser[4]. We use a 10-fold cross-validation to generate candidate lists for training. We then train the reranker by running 10 epochs of cost-augmented MIRA. The reranker uses the same features as our model, along with the tree scores obtained from the MST parser (which is a standard practice in reranking).

Experimental Details Following Koo and Collins (2010), we always first train a first-order pruner. For each word x_i, we prune away the incoming dependencies $\langle h_i, x_i \rangle$ with probability less than 0.005 times the probability of the most likely head, and limit the number of candidate heads up to 30. This gives a 99% pruning recall on the CATiB development set. The first-order model is also trained using the algorithm in Figure 4. After pruning, we tune the regularization parameter $C = \{0.1, 0.01, 0.001\}$ on development sets for different languages. Because the CoNLL datasets do not have a standard development set, we randomly select a held out of 200 sentences from the training set. We also pick the training epochs from $\{50, 100, 150\}$ which gives the best performance on the development set for each language. After tuning, the model is trained on the full training set with the selected parameters.

We apply the Random Walk-based sampling method (see Section 3.2.2) for the standard dependency parsing task. However, for the joint parsing and POS correction on the CATiB dataset we do not use the Random Walk method because the first-order features in normal parsing are no longer first-order when POS tags are also variables. Therefore, the first-order distribution is not well-defined and we only employ Gibbs sampling for simplicity. On the CATiB dataset, we restrict the sample trees to always be projective as described in Section 3.2.1. However, we do not impose this constraint for the CoNLL datasets.

6 Results

Comparison with State-of-the-art Parsers Table 2 summarizes the performance of our model and of the baselines. We first compare our model to the Turbo parser using the Turbo parser feature set. This is meant to test how our learning and inference methods compare to a dual decomposition approach. The first column in Table 2

[3] http://sourceforge.net/projects/mstparser/

[4] The MST parser is trained in projective mode for reranking because generating top-k list from second-order non-projective model is intractable.

	Our Model (UAS)		Turbo (UAS)	MST 2nd-Ord. (UAS)	Best Published UAS	Top-50 Reranker	Top-500 Reranker
	Turbo Feat.	Full Feat.					
Arabic	79.86	80.21	79.64	78.75	81.12 (Ma11)	79.03	78.91
Bulgarian	92.97	93.30	93.10	91.56	94.02 (Zh13)	92.81	-
Chinese	92.06	**92.63**	89.98	91.77	91.89 (Ma10)	92.25	-
Czech	90.62	**91.04**	90.32	87.30	90.32 (Ma13)	88.14	-
Danish	91.45	91.80	91.48	90.50	92.00 (Zh13)	90.88	90.91
Dutch	85.83	**86.47**	86.19	84.11	86.19 (Ma13)	81.01	-
English	92.79	92.94	93.22	91.54	93.22 (Ma13)	92.41	-
German	91.79	92.07	92.41	90.14	92.41 (Ma13)	91.19	-
Japanese	93.23	93.42	93.52	92.92	93.72 (Ma11)	93.40	-
Portuguese	91.82	92.41	92.69	91.08	93.03 (Ko10)	91.47	-
Slovene	86.19	86.82	86.01	83.25	86.95 (Ma11)	84.81	85.37
Spanish	**88.24**	88.21	85.59	84.33	87.96 (Zh13)	86.85	87.21
Swedish	90.48	90.71	91.14	89.05	91.62 (Zh13)	90.53	-
Turkish	76.82	77.21	76.90	74.39	77.55 (Ko10)	76.35	76.23
Average	88.87	89.23	88.72	86.86	89.33	87.92	-

Table 2: Results of our model, the Turbo parser, and the MST parser. "Best Published UAS" includes the most accurate parsers among Nivre et al. (2006), McDonald et al. (2006), Martins et al. (2010), Martins et al. (2011), Martins et al. (2013), Koo et al. (2010), Rush and Petrov (2012), Zhang and McDonald (2012) and Zhang et al. (2013). Martins et al. (2013) is the current Turbo parser. The last two columns shows UAS of the discriminative reranker.

shows the result for our model with an average of 88.87%, and the third column shows the results for the Turbo parser with an average of 88.72%. This suggests that our learning and inference procedures are as effective as the dual decomposition method in the Turbo parser.

Next, we add global features that are not used by the Turbo parser. The performance of our model is shown in the second column with an average of 89.23%. It outperforms the Turbo parser by 0.5% and achieves the best reported performance on four languages. Moreover, our model also outperforms the 88.80% average UAS reported in Martins et al. (2011), which is the top performing single parsing system (to the best of our knowledge).

Comparison with Reranking As column 6 of Table 2 shows, our model outperforms the reranker by 1.3%[5]. One possible explanation of this performance gap between the reranker and our model is the small number of candidates considered by the reranker. To test this hypothesis, we performed experiments with top-500 list for a subset of languages.[6] As column 7 shows, this increase in the list size does not change the relative performance of the reranker and our model.

Joint Parsing and POS Correction Table 3 shows the results of joint parsing and POS correction on the CATiB dataset, for our model and state-of-the-art systems. As the upper part of the table shows, the parser with corrected tags reaches 88.38% compared to the accuracy of 88.46% on the gold tags. This is a substantial increase from the parser that uses predicted tags (86.95%).

To put these numbers into perspective, the bottom part of Table 3 shows the accuracy of the best systems from the 2013 SPMRL shared task on Arabic parsing using predicted information (Seddah et al., 2013). Our system not only outperforms the best single system (Björkelund et al., 2013) by 1.4%, but it also tops the ensemble system that combines three powerful parsers: the Mate parser (Bohnet, 2010), the Easy-First parser (Goldberg and Elhadad, 2010) and the Turbo parser (Martins et al., 2013)

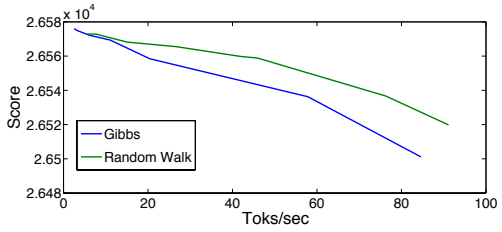
Impact of Sampling Methods We compare two sampling methods introduced in Section 3.2 with respect to their decoding efficiency. Specifically, we measure the score of the retrieved trees in testing as a function of the decoding speed, measured by the number of tokens per second. We change the temperature update rate c in order to decode with different speed. In Figure 7 we show the corresponding curves for two languages: Arabic and Chinese. We select these two languages as they correspond to two extremes in sentence length: Arabic has the longest sentences on average, while Chinese has the shortest ones. For both languages, the tree score improves over time. Given sufficient time, both sampling methods achieve the same score. However, the Random Walk-based sampler performs better when the quality is traded for speed. This result is to be expected given that each

[5]Note that the comparison is conservative because we can also add MST scores as features in our model as in reranker. With these features our model achieves an average UAS 89.28%.

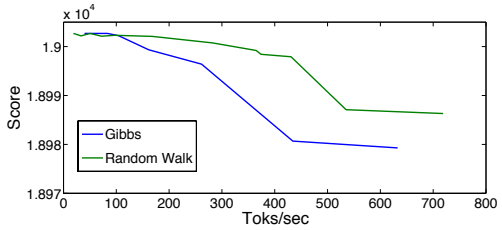
[6]We ran this experiment on 5 languages with small datasets due to the scalability issues associated with reranking top-500 list.

	Dev. Set ($\leq$ 70)		Testing Set	
	POS Acc.	UAS	POS Acc.	UAS
Gold	-	90.27	-	88.46
Predicted	96.87	88.81	96.82	86.95
POS Correction	97.72	90.08	97.49	**88.38**
CADIM	96.87	87.4-	96.82	85.78
IMS-Single	-	-	-	86.96
IMS-Ensemble	-	-	-	88.32

Table 3: Results for parsing and corrective tagging on the CATiB dataset. The upper part shows UAS of our model with gold/predicted information or POS correction. Bottom part shows UAS of the best systems in the SPMRL shared task. IMS-Single (Björkelund et al., 2013) is the best single parsing system, while IMS-Ensemble (Björkelund et al., 2013) is the best ensemble parsing system. We also show results for CADIM (Marton et al., 2013), the second best system, because we use their predicted features.



(a) Arabic



(b) Chinese

Figure 7: Total score of the predicted test trees as a function of the decoding speed, measured in the number of tokens per second.

iteration of this sampler makes multiple changes to the tree, in contrast to a single-edge change of Gibbs sampler.

The Effect of Constraints in Learning Our training method updates parameters to satisfy the pairwise constraints between (1) subsequent samples on the sampling path and (2) selected samples and the ground truth. Figure 8 shows that applying both types of constraints is consistently better than using either of them alone. Moreover, these results demonstrate that comparison between subsequent samples is more important than comparison against the gold tree.

Decoding Speed Our sampling-based parser is an anytime algorithm, and therefore its running time can be traded for performance. Figure 7 illustrates this trade-off. In the experiments reported above, we chose a conservative cooling rate and continued to sample until the score no longer changed. The parser still managed to process all the datasets in a reasonable time. For example, the time that it took to decode all the test sentences in Chinese and Arabic were 3min and 15min, respectively. Our current implementation is in Java and can be further optimized for speed.

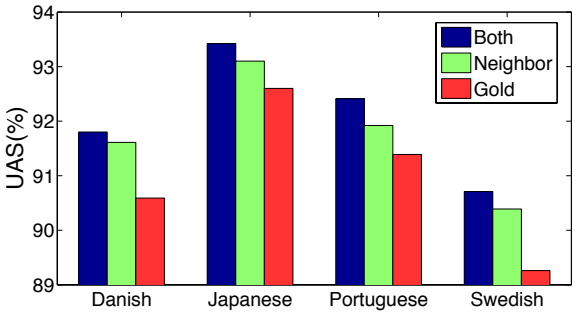


Figure 8: UAS on four languages when training with different constraints. "Neighbor" corresponds to pairwise constraints between subsequent samples, "Gold" represents constraints between a single sample and the ground truth, "Both" means applying both types of constraints.

7 Conclusions

This paper demonstrates the power of combining a simple inference procedure with a highly expressive scoring function. Our model achieves the best results on the standard dependency parsing benchmark, outperforming parsing methods with elaborate inference procedures. In addition, this framework provides simple and effective means for joint parsing and corrective tagging.

Acknowledgments

This research is developed in collaboration with the Arabic Language Technologies (ALT) group at Qatar Computing Research Institute (QCRI) within the IYAS project. The authors acknowledge the support of the MURI program (W911NF-10-1-0533, the DARPA BOLT program and the US-Israel Binational Science Foundation (BSF, Grant No 2012330). We thank the MIT NLP group and the ACL reviewers for their comments.

References

Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. 2003. An introduction to mcmc for machine learning. *Machine learning*, 50(1-2):5–43.

Anders Björkelund, Ozlem Cetinoglu, Richárd Farkas, Thomas Mueller, and Wolfgang Seeker. 2013. (re)ranking meets morphosyntax: State-of-the-art results from the SPMRL 2013 shared task. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 135–145, Seattle, Washington, USA, October. Association for Computational Linguistics.

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *COLING*, pages 89–97.

Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 149–164. Association for Computational Linguistics.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 173–180. Association for Computational Linguistics.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 175–182.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *The Journal of Machine Learning Research*, 3:951–991.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine learning*, 75(3):297–325.

Stuart Geman and Donald Geman. 1984. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):721–741.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 742–750. Association for Computational Linguistics.

Barry Haddow, Abhishek Arun, and Philipp Koehn. 2011. Samplerank training for phrase-based machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 261–271. Association for Computational Linguistics.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *ACL*, pages 586–594.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11. Association for Computational Linguistics.

Terry Koo, Alexander M Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1288–1298. Association for Computational Linguistics.

Quannan Li, Jingdong Wang, Zhuowen Tu, and David P Wipf. 2013. Fixed-point model for structured labeling. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 214–221.

André FT Martins, Noah A Smith, and Eric P Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 342–350. Association for Computational Linguistics.

André FT Martins, Noah A Smith, Eric P Xing, Pedro MQ Aguiar, and Mário AT Figueiredo. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 34–44. Association for Computational Linguistics.

André FT Martins, Noah A Smith, Pedro MQ Aguiar, and Mário AT Figueiredo. 2011. Dual decomposition with many overlapping components. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 238–249. Association for Computational Linguistics.

André FT Martins, Miguel B Almeida, and Noah A Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Yuval Marton, Nizar Habash, Owen Rambow, and Sarah Alkhulani. 2013. Spmrl13 shared task system: The cadim arabic dependency parser. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 76–80.

Ryan T McDonald and Fernando CN Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *EACL*.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 216–220. Association for Computational Linguistics.

Tetsuji Nakagawa. 2007. Multilingual dependency parsing using global features. In *EMNLP-CoNLL*, pages 952–956.

Joakim Nivre, Johan Hall, Jens Nilsson, Gülşen Eryiit, and Svetoslav Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 221–225. Association for Computational Linguistics.

Vasin Punyakanok, Dan Roth, Wen-tau Yih, and Dav Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1346. Association for Computational Linguistics.

Alexander M Rush and Slav Petrov. 2012. Vine pruning for efficient multi-pass dependency parsing. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 498–507. Association for Computational Linguistics.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, et al. 2013. Overview of the spmrl 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 146–182.

D. Sontag, A. Globerson, and T. Jaakkola. 2011. Introduction to dual decomposition for inference. In *Optimization for Machine Learning*, pages 219–254. MIT Press.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177. Association for Computational Linguistics.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Michael L. Wick, Khashayar Rohanimanesh, Kedar Bellare, Aron Culotta, and Andrew McCallum. 2011. Samplerank: Training factor graphs with atomic gradients. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pages 777–784.

David Bruce Wilson. 1996. Generating random spanning trees more quickly than the cover time. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 296–303. ACM.

Hao Zhang and Ryan McDonald. 2012. Generalized higher-order dependency parsing with cube pruning. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 320–331. Association for Computational Linguistics.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 188–193. Association for Computational Linguistics.

Hao Zhang, Liang Huang Kai Zhao, and Ryan McDonald. 2013. Online learning for inexact hypergraph search. In *Proceedings of EMNLP*.

Sparser, Better, Faster GPU Parsing

David Hall **Taylor Berg-Kirkpatrick** **John Canny** **Dan Klein**
Computer Science Division
University of California, Berkeley
{dlwh,tberg,jfc,klein}@cs.berkeley.edu

Abstract

Due to their origin in computer graphics, graphics processing units (GPUs) are highly optimized for dense problems, where the exact same operation is applied repeatedly to all data points. Natural language processing algorithms, on the other hand, are traditionally constructed in ways that exploit structural sparsity. Recently, Canny et al. (2013) presented an approach to GPU parsing that sacrifices traditional sparsity in exchange for raw computational power, obtaining a system that can compute Viterbi parses for a high-quality grammar at about 164 sentences per second on a mid-range GPU. In this work, we reintroduce sparsity to GPU parsing by adapting a coarse-to-fine pruning approach to the constraints of a GPU. The resulting system is capable of computing over 404 Viterbi parses per second—more than a 2x speedup—on the same hardware. Moreover, our approach allows us to efficiently implement less GPU-friendly minimum Bayes risk inference, improving throughput for this more accurate algorithm from only 32 sentences per second unpruned to over 190 sentences per second using pruning—nearly a 6x speedup.

1 Introduction

Because NLP models typically treat sentences independently, NLP problems have long been seen as "embarrassingly parallel" – large corpora can be processed arbitrarily fast by simply sending different sentences to different machines. However, recent trends in computer architecture, particularly the development of powerful "general purpose" GPUs, have changed the landscape even for problems that parallelize at the sentence level. First, classic single-core processors and main memory architectures are no longer getting substantially faster over time, so speed gains must now come from parallelism within a single machine. Second, compared to CPUs, GPUs devote a much larger fraction of their computational power to actual arithmetic. Since tasks like parsing boil down to repeated read-multiply-write loops, GPUs should be many times more efficient in time, power, or cost. The challenge is that GPUs are not a good fit for the kinds of sparse computations that most current CPU-based NLP algorithms rely on.

Recently, Canny et al. (2013) proposed a GPU implementation of a constituency parser that sacrifices all sparsity in exchange for the sheer horsepower that GPUs can provide. Their system uses a grammar based on the Berkeley parser (Petrov and Klein, 2007) (which is particularly amenable to GPU processing), "compiling" the grammar into a sequence of GPU kernels that are applied densely to every item in the parse chart. Together these kernels implement the Viterbi inside algorithm. On a mid-range GPU, their system can compute Viterbi derivations at 164 sentences per second on sentences of length 40 or less (see timing details below).

In this paper, we develop algorithms that can exploit sparsity on a GPU by adapting coarse-to-fine pruning to a GPU setting. On a CPU, pruning methods can give speedups of up to 100x. Such extreme speedups over a dense GPU baseline currently seem unlikely because fine-grained sparsity appears to be directly at odds with dense parallelism. However, in this paper, we present a system that finds a middle ground, where some level of sparsity can be maintained without losing the parallelism of the GPU. We use a coarse-to-fine approach as in Petrov and Klein (2007), but with only one coarse pass. Figure 1 shows an overview of the approach: we first parse densely with a coarse grammar and then parse sparsely with the

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 208–217,
Baltimore, Maryland, USA, June 23-25 2014.

fine grammar, skipping symbols that the coarse pass deemed sufficiently unlikely. Using this approach, we see a gain of more than 2x over the dense GPU implementation, resulting in overall speeds of up to 404 sentences per second. For comparison, the publicly available CPU implementation of Petrov and Klein (2007) parses approximately 7 sentences per second per core on a modern CPU.

A further drawback of the dense approach in Canny et al. (2013) is that it only computes Viterbi parses. As with other grammars with a parse/derivation distinction, the grammars of Petrov and Klein (2007) only achieve their full accuracy using minimum-Bayes-risk parsing, with improvements of over 1.5 F1 over best-derivation Viterbi parsing on the Penn Treebank (Marcus et al., 1993). To that end, we extend our coarse-to-fine GPU approach to computing marginals, along the way proposing a new way to exploit the coarse pass to avoid expensive log-domain computations in the fine pass. We then implement minimum-Bayes-risk parsing via the max recall algorithm of Goodman (1996). Without the coarse pass, the dense marginal computation is not efficient on a GPU, processing only 32 sentences per second. However, our approach allows us to process over 190 sentences per second, almost a 6x speedup.

2 A Note on Experiments

We build up our approach incrementally, with experiments interspersed throughout the paper, and summarized in Tables 1 and 2. In this paper, we focus our attention on current-generation NVIDIA GPUs. Many of the ideas described here apply to other GPUs (such as those from AMD), but some specifics will differ. All experiments are run with an NVIDIA GeForce GTX 680, a mid-range GPU that costs around $500 at time of writing. Unless otherwise noted, all experiments are conducted on sentences of length ≤ 40 words, and we estimate times based on batches of 20K sentences.[1] We should note that our experimental condition differs from that of Canny et al. (2013): they evaluate on sentences of length ≤ 30. Furthermore, they use two NVIDIA GeForce GTX *690*s—each of which is essentially a repackaging of two 680s—meaning that our system and experiments would run approximately four times faster on their hardware. (This expected 4x factor is empirically consistent with the result of running their system on our hardware.)

[1]The implementation of Canny et al. (2013) cannot handle batches so large, and so we tested it on batches of 1200 sentences. Our reimplementation is approximately the same speed for the same batch sizes. For batches of 20K sentences, we used sentences from the training set. We verified that there was no significant difference in speed for sentences from the training set and from the test set.

3 Sparsity and CPUs

One successful approach for speeding up constituency parsers has been to use coarse-to-fine inference (Charniak et al., 2006). In coarse-to-fine inference, we have a sequence of increasingly complex grammars G_ℓ. Typically, each successive grammar G_ℓ is a *refinement* of the preceding grammar $G_{\ell-1}$. That is, for each symbol A_x in the fine grammar, there is some symbol A in the coarse grammar. For instance, in a latent variable parser, the coarse grammar would have symbols like NP, VP, etc., and the fine pass would have refined symbols NP_0, NP_1, VP_4, and so on.

In coarse-to-fine inference, one applies the grammars in sequence, computing inside and outside scores. Next, one computes (max) marginals for every labeled span (A, i, j) in a sentence. These max marginals are used to compute a *pruning mask* for every span (i, j). This mask is the set of symbols allowed for that span. Then, in the next pass, one only processes rules that are licensed by the pruning mask computed at the previous level.

This approach works because a low quality coarse grammar can still reliably be used to prune many symbols from the fine chart without loss of accuracy. Petrov and Klein (2007) found that over 98% of symbols can be pruned from typical charts using a simple X-bar grammar without any loss of accuracy. Thus, the vast majority of rules can be skipped, and therefore most computation can be avoided. It is worth pointing out that although 98% of labeled spans can be skipped due to X-bar pruning, we found that only about 79% of binary rule applications can be skipped, because the unpruned symbols tend to be the ones with a larger grammar footprint.

4 GPU Architectures

Unfortunately, the standard coarse-to-fine approach does not naïvely translate to GPU architectures. GPUs work by executing thousands of threads at once, but impose the constraint that large blocks of threads must be executing the same

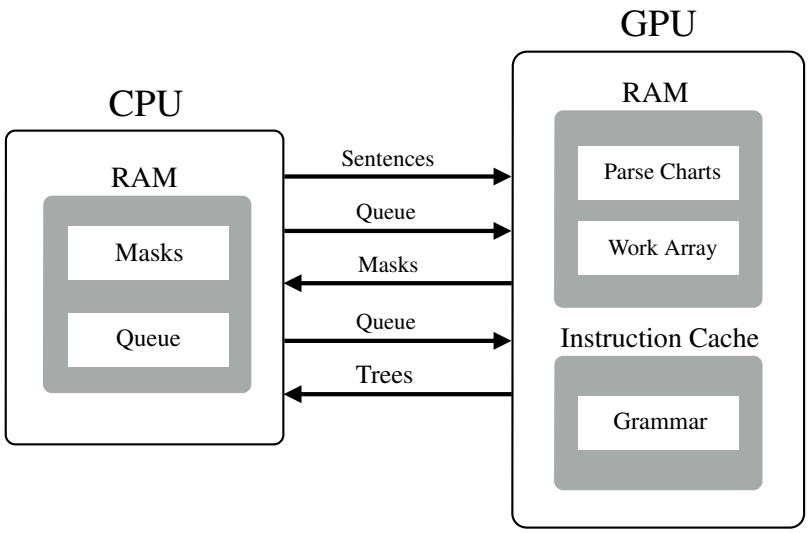


Figure 1: Overview of the architecture of our system, which is an extension of Canny et al. (2013)'s system. The GPU and CPU communicate via a work queue, which ferries parse items from the CPU to the GPU. Our system uses a coarse-to-fine approach, where the coarse pass computes a pruning mask that is used by the CPU when deciding which items to queue during the fine pass. The original system of Canny et al. (2013) only used the fine pass, with no pruning.

instructions in lockstep, differing only in their input data. Thus sparsely skipping rules and symbols will not save any work. Indeed, it may actually slow the system down. In this section, we provide an overview of GPU architectures, focusing on the details that are relevant to building an efficient parser.

The large number of threads that a GPU executes are packaged into blocks of 32 threads called *warps*. All threads in a warp must execute the same instruction at every clock cycle: if one thread takes a branch the others do not, then *all* threads in the warp must follow both code paths. This situation is called *warp divergence*. Because all threads execute all code paths that any thread takes, time can only be saved if an entire warp agrees to skip any particular branch.

NVIDIA GPUs have 8-15 processors called *streaming multi-processors* or SMs.[2] Each SM can process up to 48 different warps at a time: it interleaves the execution of each warp, so that when one warp is stalled another warp can execute. Unlike threads within a single warp, the 48 warps do not have to execute the same instructions. However, the memory architecture is such that they will be faster if they access related memory locations.

A further consideration is that the number of registers available to a thread in a warp is rather limited compared to a CPU. On the 600 series, maximum occupancy can only be achieved if each thread uses at most 63 registers (Nvidia, 2008).[3] Registers are many times faster than variables located in thread-local memory, which is actually the same speed as global memory.

5 Anatomy of a Dense GPU Parser

This architecture environment puts very different constraints on parsing algorithms from a CPU environment. Canny et al. (2013) proposed an implementation of a PCFG parser that sacrifices standard sparse methods like coarse-to-fine pruning, focusing instead on maximizing the instruction and memory throughput of the parser. They assume that they are parsing many sentences at once, with throughput being more important than latency. In this section, we describe their dense algorithm, which we take as the baseline for our work; we present it in a way that sets up the changes to follow.

At the top level, the CPU and GPU communicate via a *work queue* of parse items of the form (s, i, k, j), where s is an identifier of a sentence, i is the start of a span, k is the split point, and j

[2] Older hardware (600 series or older) has 8 SMs. Newer hardware has more.

[3] A thread can use more registers than this, but the full complement of 48 warps cannot execute if too many are used.

Clustering	Pruning	Sent/Sec	Speedup
Canny et al.	–	164.0	–
Reimpl	–	192.9	1.0x
Reimpl	Empty, Coarse	185.5	0.96x
Reimpl	Labeled, Coarse	187.5	0.97x
Parent	–	158.6	0.82x
Parent	Labeled, Coarse	278.9	1.4x
Parent	Labeled, 1-split	**404.7**	**2.1x**
Parent	Labeled, 2-split	343.6	1.8x

Table 1: Performance numbers for computing Viterbi inside charts on 20,000 sentences of length $\leq$40 from the Penn Treebank. All times are measured on an NVIDIA GeForce GTX 680. 'Reimpl' is our reimplementation of their approach. Speedups are measured in reference to this reimplementation. See Section 7 for discussion of the clustering algorithms and Section 6 for a description of the pruning methods. The Canny et al. (2013) system is benchmarked on a batch size of 1200 sentences, the others on 20,000.

is the end point. The GPU takes large numbers of parse items and applies the *entire* grammar to them in parallel. These parse items are enqueued in order of increasing span size, blocking until all items of a given length are complete. This approach is diagrammed in Figure 2.

Because all rules are applied to all parse items, all threads are executing the same sequence of instructions. Thus, there is no concern of warp divergence.

5.1 Grammar Compilation

One important feature of Canny et al. (2013)'s system is *grammar compilation*. Because registers are so much faster than thread-local memory, it is critical to keep as many variables in registers as possible. One way to accomplish this is to unroll loops at compilation time. Therefore, they inlined the iteration over the grammar directly into the GPU kernels (i.e. the code itself), which allows the compiler to more effectively use all of its registers.

However, register space is limited on GPUs. Because the Berkeley grammar is so large, the compiler is not able to efficiently schedule all of the operations in the grammar, resulting in register spills. Canny et al. (2013) found they had to partition the grammar into multiple different kernels. We discuss this partitioning in more detail in Section 7. However, in short, the entire grammar G is broken into multiple clusters G_i where each rule belongs to exactly one cluster.

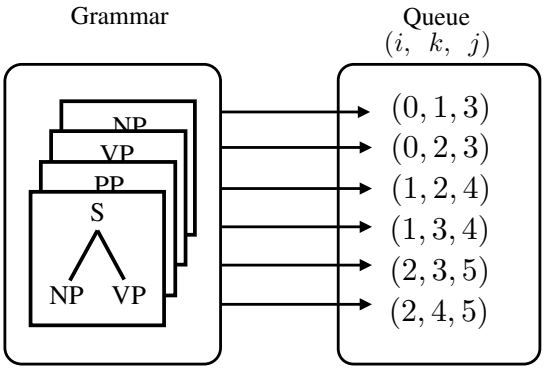


Figure 2: Schematic representation of the work queue used in Canny et al. (2013). The Viterbi inside loop for the grammar is inlined into a kernel. The kernel is applied to all items in the queue in a blockwise manner.

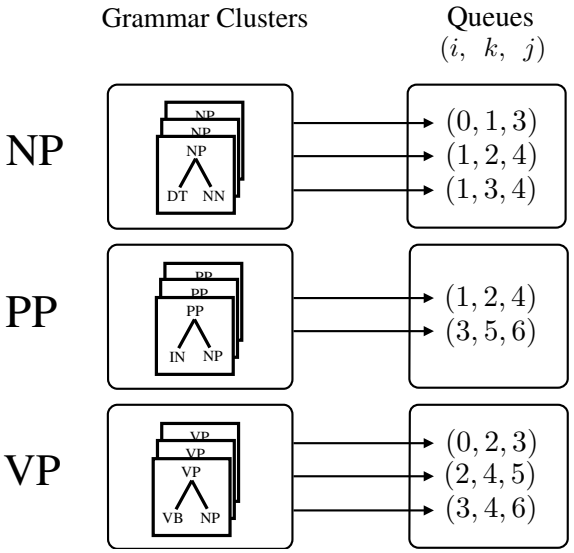


Figure 3: Schematic representation of the work queue and grammar clusters used in the fine pass of our work. Here, the rules of the grammar are clustered by their coarse parent symbol. We then have multiple work queues, with parse items only being enqueued if the span (i, j) allows that symbol in its pruning mask.

All in all, Canny et al. (2013)'s system is able to compute Viterbi charts at 164 sentences per second, for sentences up to length 40. On larger batch sizes, our reimplementation of their approach is able to achieve 193 sentences per second on the same hardware. (See Table 1.)

6 Pruning on a GPU

Now we turn to the algorithmic and architectural changes in our approach. First, consider trying to

directly apply the coarse-to-fine method sketched in Section 3 to the dense baseline described above. The natural implementation would be for each thread to check if each rule is licensed before applying it. However, we would only avoid the work of applying the rule if all threads in the warp agreed to skip it. Since each thread in the warp is processing a different span (perhaps even from a different sentence), consensus from all 32 threads on any skip would be unlikely.

Another approach would be to skip enqueuing any parse item (s, i, k, j) where the pruning mask for any of (i, j), (i, k), or (k, j) is entirely empty (i.e. all symbols are pruned in this cell by the coarse grammar). However, our experiments showed that only 40% of parse items are pruned in this manner. Because of the overhead associated with creating pruning masks and the further overhead of GPU communication, we found that this method did not actually produce any time savings at all. The result is a parsing speed of 185.5 sentences per second, as shown in Table 1 on the row labeled 'Reimpl' with 'Empty, Coarse' pruning.

Instead, we take advantage of the partitioned structure of the grammar and organize our computation around the coarse symbol set. Recall that the baseline already partitions the grammar G into rule clusters G_i to improve register sharing. (See Section 7 for more on the baseline clustering.) We create a separate work queue for each partition. We call each such queue a *labeled work queue*, and each one only queues items to which some rule in the corresponding partition applies. We call the set of coarse symbols for a partition (and therefore the corresponding labeled work queue) a *signature*.

During parsing, we only enqueue items (s, i, k, j) to a labeled queue if two conditions are met. First, the span (i, j)'s pruning mask must have a non-empty intersection with the signature of the queue. Second, the pruning mask for the children (i, k) and (k, j) must be non-empty.

Once on the GPU, parse items are processed using the same style of compiled kernel as in Canny et al. (2013). Because the entire partition (though not necessarily the entire grammar) is applied to each item in the queue, we still do not need to worry about warp divergence.

At the top level, our system first computes pruning masks with a coarse grammar. Then it processes the same sentences with the fine grammar. However, to the extent that the signatures are small, items can be selectively queued only to certain queues. This approach is diagrammed in Figure 3.

We tested our new pruning approach using an X-bar grammar as the coarse pass. The resulting speed is 187.5 sentences per second, labeled in Table 1 as row labeled 'Reimpl' with 'Labeled, Coarse' pruning. Unfortunately, this approach again does not produce a speedup relative to our reimplemented baseline. To improve upon this result, we need to consider how the grammar clustering interacts with the coarse pruning phase.

7 Grammar Clustering

Recall that the rules in the grammar are partitioned into a set of clusters, and that these clusters are further divided into subclusters. How can we best cluster and subcluster the grammar so as to maximize performance? A good clustering will group rules together that use the same symbols, since this means fewer memory accesses to read and write scores for symbols. Moreover, we would like the time spent processing each of the subclusters within a cluster to be about the same. We cannot move on to the next cluster until all threads from a cluster are finished, which means that the time a cluster takes is the amount of time taken by the longest-running subcluster. Finally, when pruning, it is best if symbols that have the same coarse projection are clustered together. That way, we are more likely to be able to skip a subcluster, since fewer distinct symbols need to be "off" for a parse item to be skipped in a given subcluster.

Canny et al. (2013) clustered *symbols* of the grammar using a sophisticated spectral clustering algorithm to obtain a permutation of the symbols. Then the rules of the grammar were laid out in a (sparse) three-dimensional tensor, with one dimension representing the parent of the rule, one representing the left child, and one representing the right child. They then split the cube into 6x2x2 contiguous "major cubes," giving a partition of the rules into 24 clusters. They then further subdivided these cubes into 2x2x2 minor cubes, giving 8 subclusters that executed in parallel. Note that the clusters induced by these major and minor cubes need not be of similar sizes; indeed, they often are not. Clustering using this method is labeled 'Reimplementation' in Table 1.

The addition of pruning introduces further considerations. First, we have a coarse grammar, with

many fewer rules and symbols. Second, we are able to skip a parse item for an entire cluster if that item's pruning mask does not intersect the cluster's signature. Spreading symbols across clusters may be inefficient: if a parse item licenses a given symbol, we will have to enqueue that item to any queue that has the symbol in its signature, no matter how many other symbols are in that cluster.

Thus, it makes sense to choose a clustering algorithm that exploits the structure introduced by the pruning masks. We use a very simple method: we cluster the rules in the grammar by coarse parent symbol. When coarse symbols are extremely unlikely (and therefore have few corresponding rules), we merge their clusters to avoid the overhead of beginning work on clusters where little work has to be done.[4] In order to subcluster, we divide up rules among subclusters so that each subcluster has the same number of active parent symbols. We found this approach to subclustering worked well in practice.

Clustering using this method is labeled 'Parent' in Table 1. Now, when we use a coarse pruning pass, we are able to parse nearly 280 sentences per second, a 70% increase in parsing performance relative to Canny et al. (2013)'s system, and nearly 50% over our reimplemented baseline.

It turns out that this simple clustering algorithm produces relatively efficient kernels even in the unpruned case. The unpruned Viterbi computations in a fine grammar using the clustering method of Canny et al. (2013) yields a speed of 193 sentences per second, whereas the same computation using coarse parent clustering has a speed of 159 sentences per second. (See Table 1.) This is not as efficient as Canny et al. (2013)'s highly tuned method, but it is still fairly fast, and much simpler to implement.

8 Pruning with Finer Grammars

The coarse to fine pruning approach of Petrov and Klein (2007) employs an X-bar grammar as its first pruning phase, but there is no reason why we cannot begin with a more complex grammar for our initial pass. As Petrov and Klein (2007) have shown, intermediate-sized Berkeley grammars prune many more symbols than the X-bar system. However, they are slower to parse with in a CPU context, and so they begin with an X-bar grammar.

Because of the overhead associated with transferring work items to GPU, using a very small grammar may not be an efficient use of the GPU's computational resources. To that end, we tried computing pruning masks with one-split and two-split Berkeley grammars. The X-bar grammar can compute pruning masks at just over 1000 sentences per second, the 1-split grammar parses 858 sentences per second, and the 2-split grammar parses 526 sentences per second.

Because parsing with these grammars is still quite fast, we tried using them as the coarse pass instead. As shown in Table 1, using a 1-split grammar as a coarse pass allows us to produce over 400 sentences per second, a full 2x improvement over our original system. Conducting a coarse pass with a 2-split grammar is somewhat slower, at a "mere" 343 sentences per second.

9 Minimum Bayes risk parsing

The Viterbi algorithm is a reasonably effective method for parsing. However, many authors have noted that parsers benefit substantially from minimum Bayes risk decoding (Goodman, 1996; Simaan, 2003; Matsuzaki et al., 2005; Titov and Henderson, 2006; Petrov and Klein, 2007). MBR algorithms for parsing do not compute the best derivation, as in Viterbi parsing, but instead the parse tree that maximizes the expected count of some figure of merit. For instance, one might want to maximize the expected number of correct constituents (Goodman, 1996), or the expected rule counts (Simaan, 2003; Petrov and Klein, 2007). MBR parsing has proven especially useful in latent variable grammars. Petrov and Klein (2007) showed that MBR trees substantially improved performance over Viterbi parses for latent variable grammars, earning up to 1.5F1.

Here, we implement the Max Recall algorithm of Goodman (1996). This algorithm maximizes the expected number of correct coarse symbols (A, i, j) with respect to the posterior distribution over parses for a sentence.

This particular MBR algorithm has the advantage that it is relatively straightforward to implement. In essence, we must compute the marginal probability of each fine-labeled span $\mu(A_x, i, j)$, and then marginalize to obtain $\mu(A, i, j)$. Then, for each span (i, j), we find the best possible split

[4] Specifically, after clustering based on the coarse parent symbol, we merge all clusters with less than 300 rules in them into one large cluster.

point k that maximizes $C(i,j) = \mu(A,i,j) + \max_k (C(i,k) + C(k,j))$. Parse extraction is then just a matter of following back pointers from the root, as in the Viterbi algorithm.

9.1 Computing marginal probabilities

The easiest way to compute marginal probabilities is to use the log space semiring rather than the Viterbi semiring, and then to run the inside and outside algorithms as before. We should expect this algorithm to be at least a factor of two slower: the outside pass performs at least as much work as the inside pass. Moreover, it typically has worse memory access patterns, leading to slower performance.

Without pruning, our approach does not handle these log domain computations well at all: we are only able to compute marginals for 32.1 sentences/second, more than a factor of 5 slower than our coarse pass. To begin, log space addition requires significantly more operations than max, which is a primitive operation on GPUs. Beyond the obvious consequence that executing more operations means more time taken, the sheer number of operations becomes too much for the compiler to handle. Because the grammars are compiled into code, the additional operations are all inlined into the kernels, producing much larger kernels. Indeed, in practice the compiler will often hang if we use the same size grammar clusters as we did for Viterbi. In practice, we found there is an effective maximum of 2000 rules per kernel using log sums, while we can use more than 10,000 rules rules in a single kernel with Viterbi.

With coarse pruning, however, we can avoid much of the increased cost associated with log domain computations. Because so many labeled spans are pruned, we are able to skip many of the grammar clusters and thus avoid many of the expensive operations. Using coarse pruning and log domain calculations, our system produces MBR trees at a rate of 130.4 sentences per second, a four-fold increase.

9.2 Scaling with the Coarse Pass

One way to avoid the expense of log domain computations is to use scaled probabilities rather than log probabilities. Scaling is one of the folk techniques that are commonly used in the NLP community, but not generally written about. Recall that floating point numbers are composed of a mantissa m and an exponent e, giving a number $f = m \cdot 2^e$. When a float underflows, the exponent becomes too low to represent the available number of bits. In scaling, floating point numbers are paired with an additional number that extends the exponent. That is, the number is represented as $f' = f \cdot \exp(s)$. Whenever f becomes either too big or too small, the number is rescaled back to a less "dangerous" range by shifting mass from the exponent e to the scaling factor s.

System	Sent/Sec	Speedup
Unpruned Log Sum MBR	32.1	–
Pruned Log Sum MBR	130.4	4.1x
Pruned Scaling MBR	**190.6**	**5.9x**
Pruned Viterbi	404.7	12.6x

Table 2: Performance numbers for computing max constituent (Goodman, 1996) trees on 20,000 sentences of length 40 or less from the Penn Treebank. For convenience, we have copied our pruned Viterbi system's result.

In practice, one scale s is used for an entire span (i,j), and all scores for that span are rescaled in concert. In our GPU system, multiple scores in any given span are being updated at the same time, which makes this dynamic rescaling tricky and expensive, especially since inter-warp communication is fairly limited.

We propose a much simpler static solution that exploits the coarse pass. In the coarse pass, we compute Viterbi inside and outside scores for every span. Because the grammar used in the coarse pass is a projection of the grammar used in the fine pass, these coarse scores correlate reasonably closely with the probabilities computed in the fine pass: If a span has a very high or very low score in the coarse pass, it typically has a similar score in the fine pass. Thus, we can use the coarse pass's inside and outside scores as the scaling values for the fine pass's scores. That is, in addition to computing a pruning mask, in the coarse pass we store the maximum inside and outside score in each span, giving two arrays of scores $s^I_{i,j}$ and $s^O_{i,j}$. Then, when applying rules in the fine pass, each fine inside score over a split span (i,k,j) is scaled to the appropriate $s^I_{i,j}$ by multiplying the score by $\exp\left(s^I_{i,k} + s^I_{k,j} - s^I_{i,j}\right)$, where $s^I_{i,k}, s^I_{k,j}, s^I_{i,j}$ are the scaling factors for the left child, right child, and parent, respectively. The outside scores are scaled analogously.

By itself, this approach works on nearly every sentence. However, scores for approximately

0.5% of sentences overflow (*sic*). Because we are summing instead of maxing scores in the fine pass, the scaling factors computed using max scores are not quite large enough, and so the rescaled inside probabilities grow too large when multiplied together. Most of this difference arises at the leaves, where the lexicon typically has more uncertainty than higher up in the tree. Therefore, in the fine pass, we normalize the inside scores at the leaves to sum to 1.0.[5] Using this slight modification, no sentences from the Treebank under- or overflow.

We know of no reason why this same trick cannot be employed in more traditional parsers, but it is especially useful here: with this static scaling, we can avoid the costly log sums without introducing any additional inter-thread communication, making the kernels much smaller and much faster. Using scaling, we are able to push our parser to 190.6 sentences/second for MBR extraction, just under half the speed of the Viterbi system.

9.3 Parsing Accuracies

It is of course important verify the correctness of our system; one easy way to do so is to examine parsing accuracy, as compared to the original Berkeley parser. We measured parsing accuracy on sentences of length ≤ 40 from section 22 of the Penn Treebank. Our Viterbi parser achieves 89.7 F1, while our MBR parser scores 91.0. These results are nearly identical to the Berkeley parsers most comparable numbers: 89.8 for Viterbi, and 90.9 for their "Max-Rule-Sum" MBR algorithm. These slight differences arise from the usual minor variation in implementation details. In particular, we use one coarse pass instead of several, and a different MBR algorithm. In addition, there are some differences in unary processing.

10 Analyzing System Performance

In this section we attempt to break down how exactly our system is spending its time. We do this in an effort to give a sense of how time is spent during computation on GPUs. These timing numbers are computed using the built-in profiling capabilities of the programming environment. As usual, profiles exhibit an observer effect, where the act of measuring the system changes the execution. Nevertheless, the general trends should more or less be preserved as compared to the unprofiled code.

System	Coarse Pass	Fine Pass
Unpruned Viterbi	–	6.4
Pruned Viterbi	1.2	1.5
Unpruned Logsum MBR	—	28.6
Pruned Scaling MBR	1.2	4.3

Table 3: Time spent in the passes of our different systems, in seconds per 1000 sentences. Pruning refers to using a 1-split grammar for the coarse pass.

To begin, we can compute the number of seconds needed to parse 1000 sentences. (We use seconds per sentence rather than sentences per second because the former measure is additive.) The results are in Table 3. In the case of pruned Viterbi, pruning reduces the amount of time spent in the fine pass by more than 4x, though half of those gains are lost to computing the pruning masks.

In Table 4, we break down the time taken by our system into individual components. As expected, binary rules account for the vast majority of the time in the unpruned Viterbi case, but much less time in the pruned case, with the total time taken for binary rules in the coarse and fine passes taking about 1/5 of the time taken by binaries in the unpruned version. Queueing, which involves copying memory around within the GPU to process the individual parse items, takes a fairly consistent amount of time in all systems. Overhead, which includes transport time between the CPU and GPU and other processing on the CPU, is relatively small for most system configurations. There is greater overhead in the scaling system, because scaling factors are copied to the CPU between the coarse and fine passes.

A final question is: how many sentences per second do we need to process to saturate the GPU's processing power? We computed Viterbi parses of successive powers of 10, from 1 to 100,000 sentences.[6] In Figure 4, we then plotted the throughput, in terms of number of sentences per second. Throughput increases through parsing 10,000 sentences, and then levels off by the time it reaches 100,000 sentences.

[5] One can instead interpret this approach as changing the scaling factors to $s^{I'}_{i,j} = s^{I}_{i,j} \cdot \prod_{i \leq k < j} \sum_A \text{inside}(A, k, k+1)$, where inside is the array of scores for the fine pass.

[6] We replicated the Treebank for the 100,000 sentences pass.

System	Coarse Pass					Fine Pass			
	Binary	Unary	Queueing	Masks	Overhead	Binary	Unary	Queueing	Overhead
Unpruned Viterbi	–	–	–	–	–	5.42	0.14	0.33	0.40
Pruned Viterbi	0.59	0.02	0.19	0.04	0.22	0.56	0.10	0.34	0.22
Pruned Scaling	0.59	0.02	0.19	0.04	0.20	1.74	0.24	0.46	0.84

Table 4: Breakdown of time spent in our different systems, in seconds per 1000 sentences. Binary and Unary refer to spent processing binary rules. Queueing refers to the amount of time used to move memory around within the GPU for processing. Overhead includes all other time, which includes communication between the GPU and the CPU.

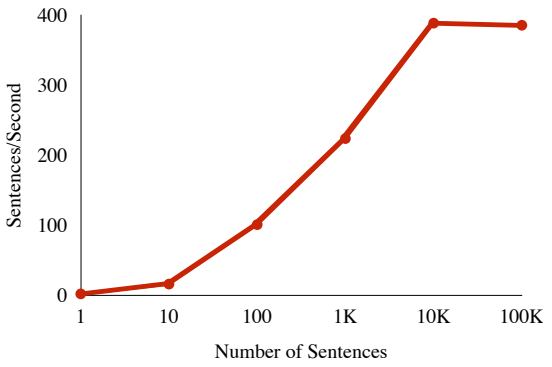


Figure 4: Plot of speeds (sentences / second) for various sizes of input corpora. The full power of the GPU parser is only reached when run on large numbers of sentences.

11 Related Work

Apart from the model of Canny et al. (2013), there have been a few attempts at using GPUs in NLP contexts before. Johnson (2011) and Yi et al. (2011) both had early attempts at porting parsing algorithms to the GPU. However, they did not demonstrate significantly increased speed over a CPU implementation. In machine translation, He et al. (2013) adapted algorithms designed for GPUs in the computational biology literature to speed up on-demand phrase table extraction.

12 Conclusion

GPUs represent a challenging opportunity for natural language processing. By carefully designing within the constraints imposed by the architecture, we have created a parser that can exploit the same kinds of sparsity that have been developed for more traditional architectures.

One of the key remaining challenges going forward is confronting the kind of lexicalized sparsity common in other NLP models. The Berkeley parser's grammars—by virtue of being unlexicalized—can be applied uniformly to all parse items. The bilexical features needed by dependency models and lexicalized constituency models are not directly amenable to acceleration using the techniques we described here. Determining how to efficiently implement these kinds of models is a promising area for new research.

Our system is available as open-source at `https://www.github.com/dlwh/puck`.

Acknowledgments

This work was partially supported by BBN under DARPA contract HR0011-12-C-0014, by a Google PhD fellowship to the first author, and an NSF fellowship to the second. We further gratefully acknowledge a hardware donation by NVIDIA Corporation.

References

John Canny, David Hall, and Dan Klein. 2013. A multi-teraflop constituency parser using GPUs. In *Proceedings of EMNLP*, pages 1898–1907, October.

Eugene Charniak, Mark Johnson, Micha Elsner, Joseph Austerweil, David Ellis, Isaac Haxton, Catherine Hill, R Shrivaths, Jeremy Moore, Michael Pozar, et al. 2006. Multilevel coarse-to-fine pcfg parsing. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 168–175. Association for Computational Linguistics.

Joshua Goodman. 1996. Parsing algorithms and metrics. In *ACL*, pages 177–183.

Hua He, Jimmy Lin, and Adam Lopez. 2013. Massively parallel suffix array queries and on-demand phrase extraction for statistical machine translation using gpus. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 325–334, Atlanta, Georgia, June. Association for Computational Linguistics.

Mark Johnson. 2011. Parsing in parallel on multiple cores and gpus. In *Proceedings of the Australasian Language Technology Association Workshop*.

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

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *ACL*, pages 75–82, Morristown, NJ, USA.

CUDA Nvidia. 2008. Programming guide.

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

Khalil Simaan. 2003. On maximizing metrics for syntactic disambiguation. In *Proceedings of IWPT*.

Ivan Titov and James Henderson. 2006. Loss minimization in parse reranking. In *Proceedings of EMNLP*, pages 560–567. Association for Computational Linguistics.

Youngmin Yi, Chao-Yue Lai, Slav Petrov, and Kurt Keutzer. 2011. Efficient parallel cky parsing on gpus. In *Proceedings of the 2011 Conference on Parsing Technologies*, Dublin, Ireland, October.

Shift-Reduce CCG Parsing with a Dependency Model

Wenduan Xu
University of Cambridge
Computer Laboratory
wx217@cam.ac.uk

Stephen Clark
University of Cambridge
Computer Laboratory
sc609@cam.ac.uk

Yue Zhang
Singapore University
of Technology and Design
yue_zhang@sutd.edu.sg

Abstract

This paper presents the first dependency model for a shift-reduce CCG parser. Modelling dependencies is desirable for a number of reasons, including handling the "spurious" ambiguity of CCG; fitting well with the theory of CCG; and optimizing for structures which are evaluated at test time. We develop a novel training technique using a dependency oracle, in which all derivations are hidden. A challenge arises from the fact that the oracle needs to keep track of exponentially many gold-standard derivations, which is solved by integrating a packed parse forest with the beam-search decoder. Standard CCGBank tests show the model achieves up to 1.05 labeled F-score improvements over three existing, competitive CCG parsing models.

1 Introduction

Combinatory Categorial Grammar (CCG; Steedman (2000)) is able to derive typed dependency structures (Hockenmaier, 2003; Clark and Curran, 2007), providing a useful approximation to the underlying predicate-argument relations of "who did what to whom". To date, CCG remains the most competitive formalism for recovering "deep" dependencies arising from many linguistic phenomena such as raising, control, extraction and coordination (Rimell et al., 2009; Nivre et al., 2010).

To achieve its expressiveness, CCG exhibits so-called "spurious" ambiguity, permitting many non-standard surface derivations which ease the recovery of certain dependencies, especially those arising from type-raising and composition. But this raises the question of what is the most suitable model for CCG: *should we model the derivations, the dependencies, or both?* The choice for some existing parsers (Hockenmaier, 2003; Clark and Curran, 2007) is to model derivations directly, restricting the gold-standard to be the normal-form derivations (Eisner, 1996) from CCGBank (Hockenmaier and Steedman, 2007).

Modelling dependencies, as a proxy for the semantic interpretation, fits well with the theory of CCG, in which Steedman (2000) argues that the derivation is merely a "trace" of the underlying syntactic process, and that the structure which is built, and predicated over when applying constraints on grammaticality, is the semantic interpretation. The early dependency model of Clark et al. (2002), in which model features were defined over *only* dependency structures, was partly motivated by these theoretical observations.

More generally, dependency models are desirable for a number of reasons. First, modelling dependencies provides an elegant solution to the spurious ambiguity problem (Clark and Curran, 2007). Second, obtaining training data for dependencies is likely to be easier than for syntactic derivations, especially for incomplete data (Schneider et al., 2013). Clark and Curran (2006) show how the dependency model from Clark and Curran (2007) extends naturally to the partial-training case, and also how to obtain dependency data cheaply from gold-standard lexical category sequences alone. And third, it has been argued that dependencies are an ideal representation for parser evaluation, especially for CCG (Briscoe and Carroll, 2006; Clark and Hockenmaier, 2002), and so optimizing for dependency recovery makes sense from an evaluation perspective.

In this paper, we fill a gap in the literature by developing the first dependency model for a shift-reduce CCG parser. Shift-reduce parsing applies naturally to CCG (Zhang and Clark, 2011), and the left-to-right, incremental nature of the decoding fits with CCG's cognitive claims. The discriminative model is global and trained with the structured perceptron. The decoder is based on beam-search

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 218–227,
Baltimore, Maryland, USA, June 23-25 2014.

(Zhang and Clark, 2008) with the advantage of linear-time decoding (Goldberg et al., 2013).

A main contribution of the paper is a novel technique for training the parser using a dependency oracle, in which all derivations are hidden. A challenge arises from the potentially exponential number of derivations leading to a gold-standard dependency structure, which the oracle needs to keep track of. Our solution is an integration of a packed parse *forest*, which efficiently stores all the derivations, with the beam-search decoder *at training time*. The derivations are not explicitly part of the data, since the forest is built from the gold-standard dependencies. We also show how perceptron learning with beam-search (Collins and Roark, 2004) can be extended to handle the additional ambiguity, by adapting the "violation-fixing" perceptron of Huang et al. (2012).

Results on the standard CCGBank tests show that our parser achieves absolute labeled F-score gains of up to 0.5 over the shift-reduce parser of Zhang and Clark (2011); and up to 1.05 and 0.64 over the normal-form and hybrid models of Clark and Curran (2007), respectively.

2 Shift-Reduce with Beam-Search

This section describes how shift-reduce techniques can be applied to CCG, following Zhang and Clark (2011). First we describe the deterministic process which a parser would follow when tracing out a single, correct derivation; then we describe how a model of normal-form derivations — or, more accurately, a sequence of shift-reduce actions leading to a normal-form derivation — can be used with beam-search to develop a non-deterministic parser which selects the highest scoring sequence of actions. Note this section only describes a normal-form derivation model for shift-reduce parsing. Section 3 explains how we extend the approach to dependency models.

The shift-reduce algorithm adapted to CCG is similar to that of shift-reduce dependency parsing (Yamada and Matsumoto, 2003; Nivre and McDonald, 2008; Zhang and Clark, 2008; Huang and Sagae, 2010). Following Zhang and Clark (2011), we define each item in the parser as a pair $\langle s, q \rangle$, where q is a queue of remaining input, consisting of words and a set of possible lexical categories for each word (with q_0 being the front word), and s is the stack that holds subtrees $s_0, s_1, ...$ (with s_0 at the top). Subtrees on the stack are partial derivations that have been built as part of the shift-reduce process. SHIFT, REDUCE and UNARY are the three types of actions that can be applied to an item. A SHIFT action shifts one of the lexical categories of q_0 onto the stack. A REDUCE action combines s_0 and s_1 according to a CCG combinatory rule, producing a new category on the top of the stack. A UNARY action applies either a type-raising or type-changing rule to the stack-top category s_0.[1]

step	stack $(s_n, ..., s_1, s_0)$	queue $(q_0, q_1, ..., q_m)$	action
0		Mr. President visited Paris	
1	N/N	President visited Paris	SHIFT
2	$N/N\ N$	visited Paris	SHIFT
3	N	visited Paris	REDUCE
4	NP	visited Paris	UNARY
5	$NP\ (S[dcl]\backslash NP)/NP$	Paris	SHIFT
6	$NP\ (S[dcl]\backslash NP)/NP\ N$		SHIFT
7	$NP\ (S[dcl]\backslash NP)/NP\ NP$		UNARY
8	$NP\ S[dcl]\backslash NP$		REDUCE
9	$S[dcl]$		REDUCE

Figure 1: Deterministic example of shift-reduce CCG parsing (lexical categories omitted on queue).

Figure 1 shows a deterministic example for the sentence *Mr. President visited Paris*, giving a single sequence of shift-reduce actions which produces a correct derivation (i.e. one producing the correct set of dependencies). Starting with the initial item $\langle s, q \rangle_0$ (row 0), which has an empty stack and a full queue, a total of nine actions are applied to produce the complete derivation.

Applying beam-search to a statistical shift-reduce parser is a straightforward extension to the deterministic example. At each step, a beam is used to store the top-k highest-scoring items, resulting from expanding all items in the previous beam. An item becomes a candidate output once it has an empty queue, and the parser keeps track of the highest scored candidate output and returns the best one as the final output. Compared with greedy local-search (Nivre and Scholz, 2004), the use of a beam allows the parser to explore a larger search space and delay difficult ambiguity-resolving decisions by considering multiple items in parallel.

We refer to the shift-reduce model of Zhang and Clark (2011) as the normal-form model, where the oracle for each sentence specifies a unique sequence of gold-standard actions which produces the corresponding normal-form derivation. No dependency structures are involved at training and test time, except for evaluation. In the next section, we describe a dependency oracle which considers all sequences of actions producing a gold-standard dependency structure to be correct.

[1] See Hockenmaier (2003) and Clark and Curran (2007) for a description of CCG rules.

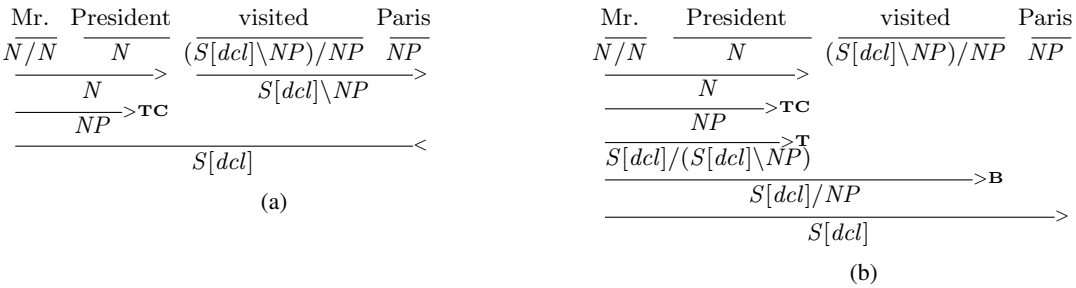


Figure 2: Two derivations leading to the same dependency structure. **TC** denotes type-changing.

3 The Dependency Model

Categories in CCG are either basic (such as NP and PP) or complex (such as $(S[dcl]\backslash NP)/NP$). Each complex category in the lexicon defines one or more predicate-argument relations, which can be realized as a predicate-argument dependency when the corresponding argument slot is consumed. For example, the transitive verb category above defines two relations: one for the subject NP and one for the object NP. In this paper a CCG predicate-argument dependency is a 4-tuple: $\langle h_f, f, s, h_a \rangle$ where h_f is the lexical item of the lexical category expressing the relation; f is the lexical category; s is the argument slot; and h_a is the head word of the argument. Since the lexical items in a dependency are indexed by their sentence positions, all dependencies for a sentence form a set, which is referred to as a CCG *dependency structure*. Clark and Curran (2007) contains a detailed description of dependency structures.

Fig. 2 shows an example demonstrating spurious ambiguity in relation to a CCG dependency structure. In both derivations, the first two lexical categories are combined using forward application ($>$) and the following dependency is realized: $\langle \textit{Mr.}, N/N_1, 1, \textit{President} \rangle$. In the normal-form derivation (a), the dependency $\langle \textit{visited}, (S\backslash NP_1)/NP_2, 2, \textit{Paris} \rangle$ is created by combining the transitive verb category with the object NP using forward application. One final dependency, $\langle \textit{visited}, (S\backslash NP_1)/NP_2, 1, \textit{President} \rangle$, is realized when the root node $S[dcl]$ is produced through backward application ($<$).

Fig. 2(b) shows a non-normal-form derivation which uses type-raising (**T**) and composition (**B**) (which are not required to derive the correct dependency structure). In this alternative derivation, the dependency $\langle \textit{visited}, (S\backslash NP_1)/NP_2, 1, \textit{President} \rangle$ is realized using forward composition (**B**), and $\langle \textit{visited}, (S\backslash NP_1)/NP_2, 2, \textit{Paris} \rangle$ is realized when the $S[dcl]$ root is produced.

The chart-based dependency model of Clark and Curran (2007) treats all derivations as hidden, and defines a probabilistic model for a dependency structure by summing probabilities of all derivations leading to a particular structure. Features are defined over both derivations and CCG predicate-argument dependencies. We follow a similar approach, but rather than define a probabilistic model (which requires summing), we define a linear model over sequences of shift-reduce actions, as for the normal-form shift-reduce model. However, the difference compared to the normal-form model is that we do not assume a single gold-standard sequence of actions.

Similar to Goldberg and Nivre (2012), we define an *oracle* which determines, for a gold-standard dependency structure, G, what the valid transition sequences are (i.e. those sequences corresponding to derivations leading to G). More specifically, the oracle can determine, given G and an item $\langle s, q \rangle$, what the valid actions are for that item (i.e. what actions can potentially lead to G, starting with $\langle s, q \rangle$ and the dependencies already built on s). However, there can be exponentially many valid action sequences for G, which we represent efficiently using a packed parse forest. We show how the forest can be used, during beam-search decoding, to determine the valid actions for a parse item (Section 3.2). We also show, in Section 3.3, how perceptron training with early-update (Collins and Roark, 2004) can be used in this setting.

3.1 The Oracle Forest

A CCG parse forest efficiently represents an exponential number of derivations. Following Clark and Curran (2007) (which builds on Miyao and Tsujii (2002)), and using the same notation, we define a CCG parse forest Φ as a tuple $\langle C, D, R, \gamma, \delta \rangle$, where C is a set of conjunctive

Algorithm 1 (Clark and Curran, 2007)

Input: A packed forest $\langle C, D, R, \gamma, \delta \rangle$, with $dmax(c)$ and $dmax(d)$ already computed

1: **function** MAIN
2: **for each** $d_r \in R$ s.t. $dmax(d_r) = |G|$ **do**
3: MARK(d_r)
4: **procedure** MARK(d)
5: mark d as a correct node
6: **for each** $c \in \gamma(d)$ **do**
7: **if** $dmax(c) == dmax(d)$ **then**
8: mark c as a correct node
9: **for each** $d' \in \delta(c)$ **do**
10: **if** d' has not been visited **then**
11: MARK(d')

nodes and D is a set of disjunctive nodes.[2] Conjunctive nodes are individual CCG categories in Φ, and are either obtained from the lexicon, or by combining two disjunctive nodes using a CCG rule, or by applying a unary rule to a disjunctive node. Disjunctive nodes are equivalence classes of conjunctive nodes. Two conjunctive nodes are equivalent iff they have the same category, head and unfilled dependencies (i.e. they will lead to the same derivation, and produce the same dependencies, in any future parsing). $R \subseteq D$ is a set of root disjunctive nodes. $\gamma : D \rightarrow 2^C$ is the conjunctive child function and $\delta : C \rightarrow 2^D$ is the disjunctive child function. The former returns the set of all conjunctive nodes of a disjunctive node, and the latter returns the disjunctive child nodes of a conjunctive node.

The dependency model requires all the conjunctive and disjunctive nodes of Φ that are part of the derivations leading to a gold-standard dependency structure G. We refer to such derivations as *correct* derivations and the packed forest containing all these derivations as the *oracle forest*, denoted as Φ_G, which is a subset of Φ. It is prohibitive to enumerate all correct derivations, but it is possible to identify, from Φ, all the conjunctive and disjunctive nodes that are part of Φ_G. Clark and Curran (2007) gives an algorithm for doing so, which we use here. The main intuition behind the algorithm is that a gold-standard dependency structure decomposes over derivations; thus gold-standard dependencies realized at conjunctive nodes can be counted when Φ is built, and all nodes that are part of Φ_G can then be marked out of Φ by traversing it top-down. A key idea in understanding the algorithm is that dependencies are created when disjunctive nodes are combined, and hence are associated with, or "live on", conjunctive nodes in the forest.

Following Clark and Curran (2007), we also define the following three values, where the first decomposes only over local rule productions, while the other two decompose over derivations:

$$cdeps(c) = \begin{cases} * & \text{if } \exists\, \tau \in deps(c), \tau \notin G \\ |deps(c)| & \text{otherwise} \end{cases}$$

$$dmax(c) = \begin{cases} * & \text{if } cdeps(c) == * \\ * & \text{if } dmax(d) == * \text{ for some } d \in \delta(c) \\ \sum_{d \in \delta(c)} dmax(d) + cdeps(c) & \text{otherwise} \end{cases}$$

$$dmax(d) = \max\{dmax(c) \mid c \in \gamma(d)\}$$

$deps(c)$ is the set of all dependencies on conjunctive node c, and $cdeps(c)$ counts the number of *correct* dependencies on c. $dmax(c)$ is the maximum number of correct dependencies over any sub-derivation headed by c and is calculated recursively; $dmax(d)$ returns the same value for a disjunctive node. In all cases, a special value $*$ indicates the presence of incorrect dependencies. To obtain the oracle forest, we first pre-compute $dmax(c)$ and $dmax(d)$ for all d and c in Φ when Φ is built using CKY, which are then used by Algorithm 1 to identify all the conjunctive and disjunctive nodes in Φ_G.

3.2 The Dependency Oracle Algorithm

We observe that the canonical shift-reduce algorithm (as demonstrated in Fig. 1) applied to a single parse tree exactly resembles bottom-up *post*-order traversal of that tree. As an example, consider the derivation in Fig. 2a, where the corresponding sequence of actions is: `sh` N/N, `sh` N, `re` N, `un` NP, `sh` $(S[dcl]\backslash NP)/NP$, `sh` NP, `re` $S[dcl]\backslash NP$, `re` $S[dcl]$.[3] The order of traversal is left-child, right-child and parent. For a single parse, the corresponding shift-reduce action sequence is unique, and for a given item this canonical order restricts the possible derivations that can be formed using further actions. We now extend this observation to the more general case of an oracle forest, where there may be more than one gold-standard action for a given item.

Definition 1. Given a gold-standard dependency

[2] Under the hypergraph framework (Gallo et al., 1993; Huang and Chiang, 2005), a conjunctive node corresponds to a hyperedge and a disjunctive node corresponds to the head of a hyperedge or hyperedge bundle.

[3] The derivation is "upside down", following the convention used for CCG, where the root is $S[dcl]$. We use `sh`, `re` and `un` to denote the three types of shift-reduce action.

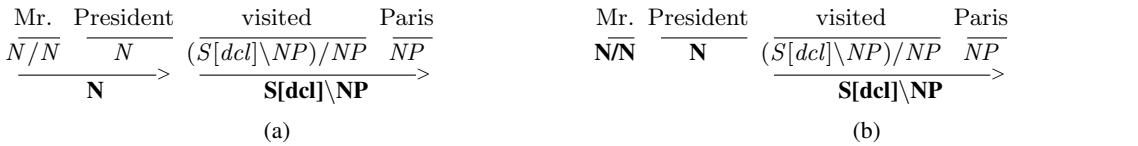


Figure 3: Example subtrees on two stacks, with two subtrees in (a) and three in (b); roots of subtrees are in bold.

structure G, an oracle forest Φ_G, and an item $\langle s, q\rangle$, we say s is a **realization** of G, denoted $s \simeq G$, if $|s| = 1$, q is empty and the single derivation on s is correct. If $|s| > 0$ and the subtrees on s can lead to a correct derivation in Φ_G using further actions, we say s is a **partial-realization** of G, denoted as $s \sim G$. And we define $s \sim G$ for $|s| = 0$.

As an example, assume that Φ_G contains only the derivation in Fig. 2a; then a stack containing the two subtrees in Fig. 3a is a partial-realization, while a stack containing the three subtrees in Fig. 3b is not. Note that each of the three subtrees in Fig. 3b is present in Φ_G; however, these subtrees cannot be combined into the single correct derivation, since the correct sequence of shift-reduce actions must first combine the lexical categories for *Mr.* and *President* before shifting the lexical category for *visited*.

We denote an action as a pair (x, c), where $x \in \{\text{SHIFT}, \text{REDUCE}, \text{UNARY}\}$ and c is the root of the subtree resulting from that action. For all three types of actions, c also corresponds to a unique conjunctive node in the *complete* forest Φ; and we use c_{s_i} to denote the conjunctive node in Φ corresponding to subtree s_i on the stack. Let $\langle s', q'\rangle = \langle s, q\rangle \circ (x, c)$ be the resulting item from applying the action (x, c) to $\langle s, q\rangle$; and let the set of all possible actions for $\langle s, q\rangle$ be $\mathcal{X}_{\langle s,q\rangle} = \{(x, c) \mid (x, c) \text{ is applicable to } \langle s, q\rangle\}$.

Definition 2. Given Φ_G and an item $\langle s, q\rangle$ s.t. $s \sim G$, we say an applicable action (x, c) for the item is **valid** iff $s' \sim G$ or $s' \simeq G$, where $\langle s', q'\rangle = \langle s, q\rangle \circ (x, c)$.

Definition 3. Given Φ_G, the dependency oracle function f_d is defined as:

$$f_d(\langle s, q\rangle, (x, c), \Phi_G) = \begin{cases} \textit{true} & \text{if } s' \sim G \text{ or } s' \simeq G \\ \textit{false} & \text{otherwise} \end{cases}$$

where $(x, c) \in \mathcal{X}_{\langle s,q\rangle}$ and $\langle s', q'\rangle = \langle s, q\rangle \circ (x, c)$.

The pseudocode in Algorithm 2 implements f_d. It determines, for a given item, whether an applicable action is valid in Φ_G.

It is trivial to determine the validity of a SHIFT action for the initial item, $\langle s, q\rangle_0$, since the SHIFT action is valid iff its category matches the gold-standard lexical category of the first word in the sentence. For any subsequent SHIFT action (SHIFT, c) to be valid, the *necessary* condition is $c \equiv c_{lex_0}$, where c_{lex_0} denotes the gold-standard lexical category of the front word in the queue, q_0 (line 3). However, this condition is not sufficient; a counterexample is the case where all the gold-standard lexical categories for the sentence in Figure 2 are shifted in succession. Hence, in general, the conditions under which an action is valid are more complex than the trivial case above.

First, suppose there is only one correct derivation in Φ_G. A SHIFT action $(\text{SHIFT}, c_{lex_0})$ is valid whenever c_{s_0} (the conjunctive node in Φ_G corresponding to the subtree s_0 on the stack) and c_{lex_0} (the conjunctive node in Φ_G corresponding to the next gold-standard lexical category from the queue) are both dominated by the conjunctive node parent p of c_{s_0} in Φ_G.[4] A REDUCE action (REDUCE, c) is valid if c matches the category of the conjunctive node parent of c_{s_0} and c_{s_1} in Φ_G. A UNARY action (UNARY, c) is valid if c matches the conjunctive node parent of c_{s_0} in Φ_G. We now generalize the case where Φ_G contains a single correct parse to the case of an oracle forest, where each parent p is replaced by a set of conjunctive nodes in Φ_G.

Definition 4. The **left parent set** $p_L(c)$ of conjunctive node $c \in \Phi_G$ is the set of all parent conjunctive nodes of c in Φ_G, which have the disjunctive node d containing c (i.e. $c \in \gamma(d)$) as a left child.

Definition 5. The **ancestor set** $\mathcal{A}(c)$ of conjunctive node $c \in \Phi_G$ is the set of all reachable ancestor conjunctive nodes of c in Φ_G.

Definition 6. Given an item $\langle s, q\rangle$, if $|s| = 1$ we say s is a **frontier stack**.

[4] Strictly speaking, the conjunctive node parent is a parent of the disjunctive node containing the conjunctive node c_{s_0}. We will continue to use this shorthand for parents of conjunctive nodes throughout the paper.

Algorithm 2 The Dependency Oracle Function f_d

Input: Φ_G, an item $\langle s, q\rangle$ s.t. $s \sim G$, $(x, c) \in \mathcal{X}_{\langle s,q\rangle}$
Let s' be the stack of $\langle s', q'\rangle = \langle s, q\rangle \circ (x, c)$

function MAIN($\langle s, q\rangle$, (x, c), Φ_G)
  if x is SHIFT **then**
    if $c \not\equiv c_{lex_0}$ **then** ▷ c not gold lexical category
      return false
    else if $c \equiv c_{lex_0}$ and $|s| = 0$ **then** ▷ the initial item
      return true
    else if $c \equiv c_{lex_0}$ and $|s| \neq 0$ **then**
      compute $\mathcal{R}(c_{s'_1}, \mathbf{c}_{s'_0})$
      return $\mathcal{R}(c_{s'_1}, \mathbf{c}_{s'_0}) \neq \emptyset$

  if x is REDUCE **then** ▷ s is non-frontier
    if $c \in \mathcal{R}(c_{s_1}, \mathbf{c}_{s_0})$ **then**
      compute $\mathcal{R}(c_{s'_1}, \mathbf{c}_{s'_0})$
      return true
    else return false

  if x is UNARY **then**
    if $|s| = 1$ **then** ▷ s is frontier
      return $c \in \Phi_G$
    if $|s| \neq 1$ **and** $c \in \Phi_G$ **then** ▷ s is non-frontier
      compute $\mathcal{R}(c_{s'_1}, \mathbf{c}_{s'_0})$
      return $\mathcal{R}(c_{s'_1}, \mathbf{c}_{s'_0}) \neq \emptyset$

A key to defining the dependency oracle function is the notion of a **shared ancestor set**. Intuitively, shared ancestor sets are built up through shift actions, and contain sets of nodes which can potentially become the results of reduce or unary actions. A further intuition is that shared ancestor sets define the space of possible correct derivations, and nodes in these sets are "ticked off" when reduce and unary actions are applied, as a single correct derivation is built through the shift-reduce process (corresponding to a bottom-up post-order traversal of the derivation). The following definition shows how the dependency oracle function builds shared ancestor sets for each action type.

Definition 7. Let $\langle s, q\rangle$ be an item and let $\langle s', q'\rangle = \langle s, q\rangle \circ (x, c)$. We define the **shared ancestor set** $\mathcal{R}(c_{s'_1}, \mathbf{c}_{s'_0})$ of $\mathbf{c}_{s'_0}$, after applying action (x, c), as:

- $\{c' \mid c' \in p_L(c_{s_0}) \cap \mathcal{A}(c)\}$, if s is frontier and $x =$ SHIFT
- $\{c' \mid c' \in p_L(c_{s_0}) \cap \mathcal{A}(c)$ and there is some $c'' \in \mathcal{R}(c_{s_1}, \mathbf{c}_{s_0})$ s.t. $c'' \in \mathcal{A}(c')\}$, if s is non-frontier and $x =$ SHIFT
- $\{c' \mid c' \in \mathcal{R}(c_{s_2}, \mathbf{c}_{s_1}) \cap \mathcal{A}(c)\}$, if $x =$ REDUCE
- $\{c' \mid c' \in \mathcal{R}(c_{s_1}, \mathbf{c}_{s_0}) \cap \mathcal{A}(c)\}$, if s is non-frontier and $x =$ UNARY
- $\mathcal{R}(\epsilon, \mathbf{c}^0_{s_0}) = \emptyset$ where $\mathbf{c}^0_{s_0}$ is the conjunctive node corresponding to the gold-standard lexical category of the first word in the sentence (ϵ is a dummy symbol indicating the bottom of stack).

The base case for Definition 7 is when the gold-standard lexical category of the first word in the sentence has been shifted, which creates an empty shared ancestor set. Furthermore, the shared ancestor set is always empty when the stack is a frontier stack.

The dependency oracle algorithm checks the validity of applicable actions. A SHIFT action is valid if $\mathcal{R}(c_{s'_1}, \mathbf{c}_{s'_0}) \neq \emptyset$ for the resulting stack s'. A valid REDUCE action consumes s_1 and s_0. For the new node, its shared ancestor set is the subset of the conjunctive nodes in $\mathcal{R}(c_{s_2}, \mathbf{c}_{s_1})$ which dominate the resulting conjunctive node of a valid REDUCE action. The UNARY case for a frontier stack is trivial: any UNARY action applicable to s in Φ_G is valid. For a non-frontier stack, the UNARY case is similar to REDUCE except the resulting shared ancestor set is a subset of $\mathcal{R}(c_{s_1}, \mathbf{c}_{s_0})$.

We now turn to the problem of finding the shared ancestor sets. In practice, we do not do this by traversing Φ_G top-down from the conjunctive nodes in $p_L(c_{s_0})$ on-the-fly to find each member of $\mathcal{R}$. Instead, when we build Φ_G in bottom-up topological order, we pre-compute the set of reachable disjunctive nodes of each conjunctive node c in Φ_G as:

$$\mathcal{D}(c) = \delta(c) \cup (\cup_{c' \in \gamma(d), d \in \delta(c)}(\mathcal{D}(c')))$$

Each $\mathcal{D}$ is implemented as a hash map, which allows us to test the membership of one potential conjunctive node in $\mathcal{O}(1)$ time. For example, a conjunctive node $c \in p_L(c_{s_0})$ is reachable from c_{lex_0} if there is a *disjunctive node* $d \in \mathcal{D}(c)$ s.t. $c_{lex_0} \in \gamma(d)$. With this implementation, the complexity of checking each valid SHIFT action is then $\mathcal{O}(|p_L(c_{s_0})|)$.

3.3 Training

We use the averaged perceptron (Collins, 2002) to train a global linear model and score each action. The normal-form model of Zhang and Clark (2011) uses an early update mechanism (Collins and Roark, 2004), where decoding is stopped to update model weights whenever the single gold action falls outside the beam. In our parser, there can be multiple gold items in a beam. One option would be to apply early update whenever at least

Algorithm 3 Dependency Model Training

Input: (y, G) and beam size k
1: $\mathbf{w} \leftarrow \mathbf{0}$; $\mathcal{B}_0 \leftarrow \emptyset$; $i \leftarrow 0$
2: $\mathcal{B}_0$.push($\langle s, q \rangle_0$) ▷ the initial item
3: $cand \leftarrow \emptyset$ ▷ candidate output priority queue
4: $gold \leftarrow \emptyset$ ▷ gold output priority queue
5: **while** $\mathcal{B}_i \neq \emptyset$ **do**
6: **for each** $\langle s, q \rangle \in \mathcal{B}_i$ **do**
7: **if** $|q| = 0$ **then** ▷ candidate output
8: $cand$.push($\langle s, q \rangle$)
9: **if** $s \simeq G$ **then** ▷ s is a realization of G
10: $gold$.push($\langle s, q \rangle$)
11: expand $\langle s, q \rangle$ into $\mathcal{B}_{i+1}$
12: $\mathcal{B}_{i+1} \leftarrow \mathcal{B}_{i+1}[1:k]$ ▷ apply beam
13: **if** $\Pi_G \neq \emptyset, \Pi_G \cap \mathcal{B}_{i+1} = \emptyset$ **and** $cand[0] \not\simeq G$ **then**
14: $\mathbf{w} \leftarrow \mathbf{w} + \phi(\Pi_G[0]) - \phi(\mathcal{B}_{i+1}[0])$ ▷ early update
15: **return**
16: $i \leftarrow i + 1$ ▷ continue to next step
17: **if** $cand[0] \not\simeq G$ **then** ▷ final update
18: $\mathbf{w} \leftarrow \mathbf{w} + \phi(gold[0]) - \phi(cand[0])$

one of these gold items falls outside the beam. However, this may not be a true violation of the gold-standard (Huang et al., 2012). Thus, we use a relaxed version of early update, in which *all* gold-standard actions must fall outside the beam before an update is performed. This update mechanism is provably correct under the violation-fixing framework of Huang et al. (2012).

Let (y, G) be a training sentence paired with its gold-standard dependency structure and let $\Pi_{\langle s,q \rangle}$ be the following set for an item $\langle s, q \rangle$:

$$\{\langle s, q \rangle \circ (x, c) \mid f_d(\langle s, q \rangle, (x, c), \Phi_G) = true\}$$

$\Pi_{\langle s,q \rangle}$ contains all *correct* items at step $i + 1$ obtained by expanding $\langle s, q \rangle$. Let the set of *all* correct items at a step $i + 1$ be:[5]

$$\Pi_G = \bigcup_{\langle s,q \rangle \in \mathcal{B}_i} \Pi_{\langle s,q \rangle}$$

Algorithm 3 shows the pseudocode for training the dependency model with early update for one input (y, G). The score of an item $\langle s, q \rangle$ is calculated as $\mathbf{w} \cdot \phi(\langle s, q \rangle)$ with respect to the current model $\mathbf{w}$, where $\phi(\langle s, q \rangle)$ is the feature vector for the item. At step i, all items are expanded and added onto the next beam $\mathcal{B}_{i+1}$, and the top-k retained. Early update is applied when all gold items first fall outside the beam, and any candidate output is incorrect (line 14). Since there are potentially many gold items, and one gold item is required for the perceptron update, a decision needs to be made regarding which gold item to update against. We choose to reward the highest scoring gold item, in line with the violation-fixing framework; and penalize the highest scoring incorrect item, using the standard perceptron update. A final update is performed if no more expansions are possible but the final output is incorrect.

[5] In Algorithm 3 we abuse notation by using $\Pi_G[0]$ to denote the highest scoring gold item in the set.

4 Experiments

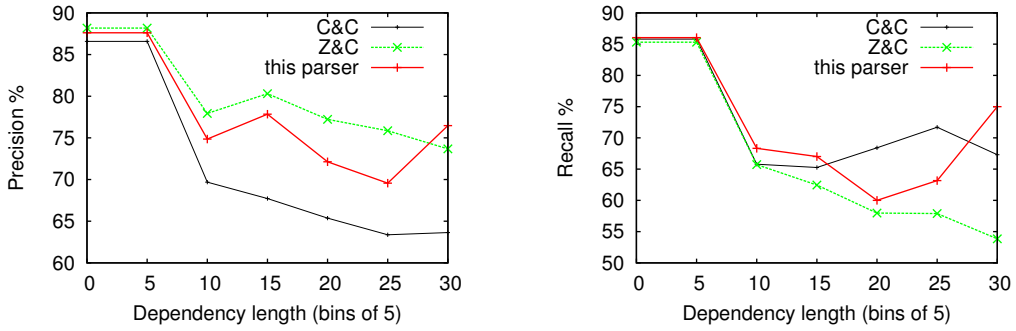
We implement our shift-reduce parser on top of the core C&C code base (Clark and Curran, 2007) and evaluate it against the shift-reduce parser of Zhang and Clark (2011) (henceforth Z&C) and the chart-based normal-form and hybrid models of Clark and Curran (2007). For all experiments, we use CCGBank with the standard split: sections 2-21 for training (39,604 sentences), section 00 for development (1,913 sentences) and section 23 (2,407 sentences) for testing.

The way that the CCG grammar is implemented in C&C has some implications for our parser. First, unlike Z&C, which uses a context-free cover (Fowler and Penn, 2010) and hence is able to use all sentences in the training data, we are only able to use 36,036 sentences. The reason is that the grammar in C&C does not have complete coverage of CCGBank, due to the fact that e.g. not all rules in CCGBank conform to the combinatory rules of CCG. Second, our parser uses the unification mechanism from C&C to output dependencies directly, and hence does not need a separate post-processing step to convert derivations into CCG dependencies, as required by Z&C.

The feature templates of our model consist of all of those in Z&C, except the ones which require lexical heads to come from either the left or right child, as such features are incompatible with the head passing mechanism used by C&C. Each Z&C template is defined over a parse item, and captures various aspects of the stack and queue context. For example, one template returns the top category on the stack plus its head word, together with the first word and its POS tag on the queue. Another template returns the second category on the stack, together with the POS tag of its head word. Every Z&C feature is defined as a pair, consisting of an instantiated context template and a parse action. In addition, we use all the CCG predicate-argument dependency features from Clark and Curran (2007), which contribute to the score of a REDUCE action when dependencies

	LP %	LR %	LF %	LSent. %	CatAcc. %	coverage %
this parser	86.29	**84.09**	**85.18**	**34.40**	92.75	100
Z&C	**87.15**	82.95	85.00	33.82	**92.77**	100
C&C (normal-form)	85.22	82.52	83.85	31.63	92.40	100
this parser	86.76	**84.90**	**85.82**	**34.72**	**93.20**	99.06 (C&C coverage)
Z&C	**87.55**	83.63	85.54	34.14	93.11	99.06 (C&C coverage)
C&C (hybrid)	–	–	85.25	–	–	99.06 (C&C coverage)
C&C (normal-form)	85.22	84.29	84.76	31.93	92.83	99.06 (C&C coverage)

Table 1: Accuracy comparison on Section 00 (auto POS).



(a) precision vs. dependency length

(b) recall vs. dependency length

Figure 4: Labeled precision and recall relative to dependency length on the development set. C&C normal-form model is used.

are realized. Detailed descriptions of all the templates in our model can be found in the respective papers. We run 20 training iterations and the resulting model contains 16.5M features with a non-zero weight.

We use 10-fold cross validation for POS tagging and supertagging the training data, and automatically assigned POS tags for all experiments. A probability cut-off value of 0.0001 for the β parameter in the supertagger is used for both training and testing. The β parameter determines how many lexical categories are assigned to each word; $\beta = 0.0001$ is a relatively small value which allows in a large number of categories, compared to the default value used in Clark and Curran (2007). For training only, if the gold-standard lexical category is not supplied by the supertagger for a particular word, it is added to the list of categories.

4.1 Results and Analysis

The beam size was tuned on the development set, and a value of 128 was found to achieve a reasonable balance of accuracy and speed; hence this value was used for all experiments. Since C&C always enforces non-fragmentary output (i.e. it can only produce spanning analyses), it fails on some sentences in the development and test sets, and thus we also evaluate on the reduced sets, following Clark and Curran (2007). Our parser does not fail on any sentences because it permits fragmentary output (those cases where there is more than one subtree left on the final stack). The results for Z&C, and the C&C normal-form and hybrid models, are taken from Zhang and Clark (2011).

Table 1 shows the accuracies of all parsers on the development set, in terms of labeled precision and recall over the predicate-argument dependencies in CCGBank. On both the full and reduced sets, our parser achieves the highest F-score. In comparison with C&C, our parser shows significant increases across all metrics, with 0.57% and 1.06% absolute F-score improvements over the hybrid and normal-form models, respectively. Another major improvement over the other two parsers is in sentence level accuracy, LSent, which measures the number of sentences for which the dependency structure is completely correct.

Table 1 also shows that our parser has improved recall over Z&C at some expense of precision. To probe this further we compare labeled precision and recall relative to dependency length, as measured by the distance between the two words in a dependency, grouped into bins of 5 values. Fig. 4 shows clearly that Z&C favors precision over recall, giving higher precision scores for almost all dependency lengths compared to our parser. In

category	LP % (o)	LP % (z)	LP % (c)	LR % (o)	LR % (z)	LR % (c)	LF % (o)	LF % (z)	LF % (c)	freq.
N/**N**	95.53	**95.77**	95.28	**95.83**	95.79	95.62	95.68	**95.78**	95.45	7288
NP/**N**	96.53	**96.70**	96.57	**97.12**	96.59	96.03	**96.83**	96.65	96.30	4101
$(NP\backslash NP)$/**NP**	81.64	**83.19**	82.17	**90.63**	89.24	88.90	85.90	**86.11**	85.40	2379
$(NP\backslash$**NP**$)/NP$	81.70	**82.53**	81.58	**88.91**	87.99	85.74	85.15	**85.17**	83.61	2174
$((S\backslash NP)\backslash(S\backslash NP))$/**NP**	**77.64**	77.60	71.94	72.97	71.58	**73.32**	**75.24**	74.47	72.63	1147
$((S\backslash NP)\backslash($**S**$\backslash$**NP**$))/NP$	75.78	**76.30**	70.92	71.27	70.60	**71.93**	**73.45**	73.34	71.42	1058
$((S[dcl]\backslash NP)$/**NP**	83.94	**85.60**	81.57	86.04	84.30	**86.37**	**84.98**	84.95	83.90	917
PP/**NP**	**77.06**	73.76	75.06	**73.63**	72.83	70.09	**75.31**	73.29	72.49	876
$((S[dcl]\backslash$**NP**$)/NP$	82.03	**85.32**	81.62	83.26	82.00	**85.55**	82.64	**83.63**	83.54	872
$((S\backslash NP)\backslash($**S**$\backslash$**NP**$))$	86.42	84.44	**86.85**	86.19	86.60	**86.73**	86.31	85.51	**86.79**	746

Table 2: Accuracy comparison on most frequent dependency types, for our parser (o), Z&C (z) and C&C hybrid model (c). Categories in bold indicate the argument slot in the relation.

	LP %	LR %	LF %	LSent. %	CatAcc. %	coverage %
our parser	87.03	**85.08**	**86.04**	**35.69**	93.10	100
Z&C	**87.43**	83.61	85.48	35.19	**93.12**	100
C&C (normal-form)	85.58	82.85	84.20	32.90	92.84	100
our parser	87.04	**85.16**	**86.09**	**35.84**	93.13	99.58 (C&C coverage)
Z&C	**87.43**	83.71	85.53	35.34	**93.15**	99.58 (C&C coverage)
C&C (hybrid)	86.17	84.74	85.45	32.92	92.98	99.58 (C&C coverage)
C&C (normal-form)	85.48	84.60	85.04	33.08	92.86	99.58 (C&C coverage)

Table 3: Accuracy comparison on section 23 (auto POS).

terms of recall (Fig. 4b), our parser outperforms Z&C over all dependency lengths, especially for longer dependencies ($x \geq 20$). When compared with C&C, the recall of the Z&C parser drops quickly for dependency lengths over 10. While our parser also suffers from this problem, it is less severe and is able to achieve higher recall at $x \geq 30$.

Table 2 compares our parser with Z&C and the C&C hybrid model, for the most frequent dependency relations. While our parser achieved lower precision than Z&C, it is more balanced and gives higher recall for all of the dependency relations except the last one, and higher F-score for over half of them.

Table 3 presents the final test results on Section 23. Again, our parser achieves the highest scores across all metrics (for both the full and reduced test sets), except for precision and lexical category assignment, where Z&C performed better.

5 Conclusion

We have presented a dependency model for a shift-reduce CCG parser, which fully aligns CCG parsing with the left-to-right, incremental nature of a shift-reduce parser. Our work is in part inspired by the dependency models of Clark and Curran (2007) and, in the use of a dependency oracle, is close in spirit to that of Goldberg and Nivre (2012). The difference is that the Goldberg and Nivre parser builds, and scores, dependency structures directly, whereas our parser uses a unification mechanism to create dependencies, and scores the CCG derivations, allowing great flexibility in terms of what dependencies can be realized. Another related work is Yu et al. (2013), which introduced a similar technique to deal with spurious ambiguity in MT. Finally, there may be potential to integrate the techniques of Auli and Lopez (2011), which currently represents the state-of-the-art in CCGBank parsing, into our parser.

Acknowledgements

We thank the anonymous reviewers for their helpful comments. Wenduan Xu is fully supported by the Carnegie Trust and receives additional funding from the Cambridge Trusts. Stephen Clark is supported by ERC Starting Grant DisCoTex (306920) and EPSRC grant EP/I037512/1. Yue Zhang is supported by Singapore MOE Tier2 grant T2MOE201301.

References

Michael Auli and Adam Lopez. 2011. A comparison of loopy belief propagation and dual decomposition for integrated CCG supertagging and parsing. In *Proc. ACL 2011*, pages 470–480, Portland, OR.

Ted Briscoe and John Carroll. 2006. Evaluating the accuracy of an unlexicalized statistical parser on the

PARC DepBank. In *Proc. of COLING/ACL*, pages 41–48, Sydney, Australia.

Stephen Clark and James R. Curran. 2006. Partial training for a lexicalized-grammar parser. In *Proc. NAACL-06*, pages 144–151, New York, USA.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Stephen Clark and Julia Hockenmaier. 2002. Evaluating a wide-coverage CCG parser. In *Proc. of the LREC 2002 Beyond Parseval Workshop*, pages 60–66, Las Palmas, Spain.

Stephen Clark, Julia Hockenmaier, and Mark Steedman. 2002. Building deep dependency structures with a wide-coverage CCG parser. In *Proc. ACL*, pages 327–334, Philadelphia, PA.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proc. of ACL*, pages 111–118, Barcelona, Spain.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proc. of EMNLP*, pages 1–8, Philadelphia, USA.

Jason Eisner. 1996. Efficient normal-form parsing for Combinatory Categorial Grammar. In *Proc. ACL*, pages 79–86, Santa Cruz, CA.

Timothy AD Fowler and Gerald Penn. 2010. Accurate context-free parsing with Combinatory Categorial Grammar. In *Proc. ACL*, pages 335–344, Uppsala, Sweden.

Giorgio Gallo, Giustino Longo, Stefano Pallottino, and Sang Nguyen. 1993. Directed hypergraphs and applications. *Discrete applied mathematics*, 42(2):177–201.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proc. COLING*, Mumbai, India.

Yoav Goldberg, Kai Zhao, and Liang Huang. 2013. Efficient implementation for beam search incremental parsers. In *Proceedings of the Short Papers of ACL*, Sofia, Bulgaria.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Julia Hockenmaier. 2003. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 53–64, Vancouver, Canada.

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proc. ACL*, pages 1077–1086, Uppsala, Sweden.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proc. NAACL*, pages 142–151, Montreal, Canada.

Yusuke Miyao and Jun'ichi Tsujii. 2002. Maximum entropy estimation for feature forests. In *Proceedings of the Human Language Technology Conference*, San Diego, CA.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proc. of ACL/HLT*, pages 950–958, Columbus, Ohio.

J. Nivre and M Scholz. 2004. Deterministic dependency parsing of English text. In *Proceedings of COLING 2004*, pages 64–70, Geneva, Switzerland.

Joakim Nivre, Laura Rimell, Ryan McDonald, and Carlos Gomez-Rodriguez. 2010. Evaluation of dependency parsers on unbounded dependencies. In *Proc. of COLING*, Beijing, China.

Laura Rimell, Stephen Clark, and Mark Steedman. 2009. Unbounded dependency recovery for parser evaluation. In *Proc. EMNLP*, pages 813–821, Edinburgh, UK.

Nathan Schneider, Brendan O'Connor, Naomi Saphra, David Bamman, Manaal Faruqui, Noah A. Smith, Chris Dyer, and Jason Baldridge. 2013. A framework for (under)specifying dependency syntax without overloading annotators. In *Proc. of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, Sofia, Bulgaria.

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press, Cambridge, Mass.

H Yamada and Y Matsumoto. 2003. Statistical dependency analysis using support vector machines. In *Proc. of IWPT*, Nancy, France.

Heng Yu, Liang Huang, Haitao Mi, and Kai Zhao. 2013. Max-violation perceptron and forced decoding for scalable mt training. In *Proc. EMNLP, Seattle, Washington, USA*.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search. In *Proc. of EMNLP*, Hawaii, USA.

Yue Zhang and Stephen Clark. 2011. Shift-reduce CCG parsing. In *Proc. ACL 2011*, pages 683–692, Portland, OR.

Less Grammar, More Features

David Hall **Greg Durrett** **Dan Klein**
Computer Science Division
University of California, Berkeley
{dlwh,gdurrett,klein}@cs.berkeley.edu

Abstract

We present a parser that relies primarily on extracting information directly from surface spans rather than on propagating information through enriched grammar structure. For example, instead of creating separate grammar symbols to mark the definiteness of an NP, our parser might instead capture the same information from the first word of the NP. Moving context out of the grammar and onto surface features can greatly simplify the structural component of the parser: because so many deep syntactic cues have surface reflexes, our system can still parse accurately with context-free backbones as minimal as X-bar grammars. Keeping the structural backbone simple and moving features to the surface also allows easy adaptation to new languages and even to new tasks. On the SPMRL 2013 multilingual constituency parsing shared task (Seddah et al., 2013), our system outperforms the top single parser system of Björkelund et al. (2013) on a range of languages. In addition, despite being designed for syntactic analysis, our system also achieves state-of-the-art numbers on the structural sentiment task of Socher et al. (2013). Finally, we show that, in both syntactic parsing and sentiment analysis, many broad linguistic trends can be captured via surface features.

1 Introduction

Naïve context-free grammars, such as those embodied by standard treebank annotations, do not parse well because their symbols have too little context to constrain their syntactic behavior. For example, *to* PPs usually attach to verbs and *of* PPs usually attach to nouns, but a context-free PP symbol can equally well attach to either. Much of the last few decades of parsing research has therefore focused on propagating contextual information from the leaves of the tree to internal nodes. For example, head lexicalization (Eisner, 1996; Collins, 1997; Charniak, 1997), structural annotation (Johnson, 1998; Klein and Manning, 2003), and state-splitting (Matsuzaki et al., 2005; Petrov et al., 2006) are all designed to take coarse symbols like PP and decorate them with additional context. The underlying reason that such propagation is even needed is that PCFG parsers score trees based on local configurations only, and any information that is not threaded through the tree becomes inaccessible to the scoring function. There have been non-local approaches as well, such as tree-substitution parsers (Bod, 1993; Sima'an, 2000), neural net parsers (Henderson, 2003), and rerankers (Collins and Koo, 2005; Charniak and Johnson, 2005; Huang, 2008). These non-local approaches can actually go even further in enriching the grammar's structural complexity by coupling larger domains in various ways, though their non-locality generally complicates inference.

In this work, we instead try to *minimize* the structural complexity of the grammar by moving as much context as possible onto local surface features. We examine the position that grammars should not propagate any information that is available from surface strings, since a discriminative parser can access that information directly. We therefore begin with a minimal grammar and iteratively augment it with rich input features that do not enrich the context-free backbone. Previous work has also used surface features in their parsers, but the focus has been on machine learning methods (Taskar et al., 2004), latent annotations (Petrov and Klein, 2008a; Petrov and Klein, 2008b), or implementation (Finkel et al., 2008).

By contrast, we investigate the extent to which

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 228–237,
Baltimore, Maryland, USA, June 23-25 2014.

we need a grammar at all. As a thought experiment, consider a parser with no grammar, which functions by independently classifying each span (i, j) of a sentence as an NP, VP, and so on, or *null* if that span is a non-constituent. For example, spans that begin with *the* might tend to be NPs, while spans that end with *of* might tend to be non-constituents. An independent classification approach is actually very viable for part-of-speech tagging (Toutanova et al., 2003), but is problematic for parsing – if nothing else, parsing comes with a structural requirement that the output be a well-formed, nested tree. Our parser uses a minimal PCFG backbone grammar to ensure a basic level of structural well-formedness, but relies mostly on features of surface spans to drive accuracy. Formally, our model is a CRF where the features factor over anchored rules of a small backbone grammar, as shown in Figure 1.

Some aspects of the parsing problem, such as the tree constraint, are clearly best captured by a PCFG. Others, such as heaviness effects, are naturally captured using surface information. The open question is whether surface features are adequate for key effects like subcategorization, which have deep definitions but regular surface reflexes (e.g. the preposition selected by a verb will often linearly follow it). Empirically, the answer seems to be yes, and our system produces strong results, e.g. up to 90.5 F1 on English parsing. Our parser is also able to generalize well across languages with little tuning: it achieves state-of-the-art results on multilingual parsing, scoring higher than the best single-parser system from the SPMRL 2013 Shared Task on a range of languages, as well as on the competition's average F1 metric.

One advantage of a system that relies on surface features and a simple grammar is that it is portable not only across languages but also across tasks to an extent. For example, Socher et al. (2013) demonstrates that sentiment analysis, which is usually approached as a flat classification task, can be viewed as tree-structured. In their work, they propagate real-valued vectors up a tree using neural tensor nets and see gains from their recursive approach. Our parser can be easily adapted to this task by replacing the X-bar grammar over treebank symbols with a grammar over the sentiment values to encode the output variables and then adding n-gram indicators to our feature set to capture the bulk of the lexical effects. When applied to this task, our system generally matches their accuracy overall and is able to outperform it on the overall sentence-level subtask.

2 Parsing Model

In order to exploit non-independent surface features of the input, we use a discriminative formulation. Our model is a conditional random field (Lafferty et al., 2001) over trees, in the same vein as Finkel et al. (2008) and Petrov and Klein (2008a). Formally, we define the probability of a tree T conditioned on a sentence $\mathbf{w}$ as

$$p(T|\mathbf{w}) \propto \exp\left(\theta^\intercal \sum_{r \in T} f(r, \mathbf{w})\right) \qquad (1)$$

where the feature domains r range over the (anchored) rules used in the tree. An anchored rule r is the conjunction of an unanchored grammar rule $\mathrm{rule}(r)$ and the start, stop, and split indexes where that rule is anchored, which we refer to as $\mathrm{span}(r)$. It is important to note that the richness of the backbone grammar is reflected in the structure of the trees T, while the features that condition directly on the input enter the equation through the anchoring $\mathrm{span}(r)$. To optimize model parameters, we use the Adagrad algorithm of Duchi et al. (2010) with L2 regularization.

We start with a simple X-bar grammar whose only symbols are NP, NP-bar, VP, and so on. Our base model has no surface features: formally, on each anchored rule r we have only an indicator of the (unanchored) rule identity, $\mathrm{rule}(r)$. Because the X-bar grammar is so minimal, this grammar does not parse very accurately, scoring just 73 F1 on the standard English Penn Treebank task.

In past work that has used tree-structured CRFs in this way, increased accuracy partially came from decorating trees T with additional annotations, giving a tree T' over a more complex symbol set. These annotations introduce additional context into the model, usually capturing linguistic intuition about the factors that influence grammaticality. For instance, we might annotate every constituent X in the tree with its parent Y, giving a tree with symbols $X[\hat{}Y]$. Finkel et al. (2008) used parent annotation, head tag annotation, and horizontal sibling annotation together in a single large grammar. In Petrov and Klein (2008a) and Petrov and Klein (2008b), these annotations were latent; they were inferred automatically during training.

Hall and Klein (2012) employed both kinds of annotations, along with lexicalized head word annotation. All of these past CRF parsers do also exploit span features, as did the structured margin parser of Taskar et al. (2004); the current work primarily differs in shifting the work from the grammar to the surface features.

The problem with rich annotations is that they increase the state space of the grammar substantially. For example, adding parent annotation can square the number of symbols, and each subsequent annotation causes a multiplicative increase in the size of the state space. Hall and Klein (2012) attempted to reduce this state space by factoring these annotations into individual components. Their approach changed the multiplicative penalty of annotation into an additive penalty, but even so their individual grammar projections are much larger than the base X-bar grammar.

In this work, we want to see how much of the expressive capability of annotations can be captured using surface evidence, with little or no annotation of the underlying grammar. To that end, we avoid annotating our trees at all, opting instead to see how far simple surface features will go in achieving a high-performance parser. We will return to the question of annotation in Section 5.

3 Surface Feature Framework

To improve the performance of our X-bar grammar, we will add a number of surface feature templates derived only from the words in the sentence. We say that an indicator is a surface property if it can be extracted without reference to the parse tree. These features can be implemented without reference to structured linguistic notions like headedness; however, we will argue that they still capture a wide range of linguistic phenomena in a data-driven way.

Throughout this and the following section, we will draw on motivating examples from the English Penn Treebank, though similar examples could be equally argued for other languages. For performance on other languages, see Section 6.

Recall that our CRF factors over anchored rules r, where each r has identity $\text{rule}(r)$ and anchoring $\text{span}(r)$. The X-bar grammar has only indicators of $\text{rule}(r)$, ignoring the anchoring. Let a *surface property* of r be an indicator function of $\text{span}(r)$ and the sentence itself. For example, the first word in a constituent is a surface property, as

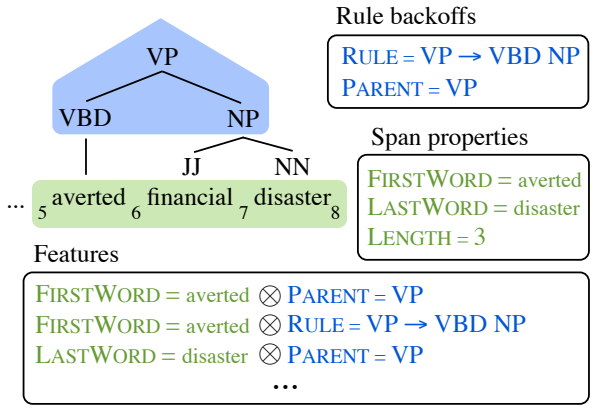


Figure 1: Features computed over the application of the rule VP → VBD NP over the anchored span *averted financial disaster* with the shown indices. Span properties are generated as described throughout Section 4; they are then conjoined with the rule and just the parent nonterminal to give the features fired over the anchored production.

is the word directly preceding the constituent. As illustrated in Figure 1, the actual features of the model are obtained by conjoining surface properties with various abstractions of the rule identity. For rule abstractions, we use two templates: the parent of the rule and the identity of the rule. The surface features are somewhat more involved, and so we introduce them incrementally.

One immediate computational and statistical issue arises from the sheer number of possible surface features. There are a great number of spans in a typical treebank; extracting features for every possible combination of span and rule is prohibitive. One simple solution is to only extract features for rule/span pairs that are actually observed in gold annotated examples during training. Because these "positive" features correspond to observed constituents, they are far less numerous than the set of all possible features extracted from all spans. As far as we can tell, all past CRF parsers have used "positive" features only.

However, negative features—features that are not observed in any tree—are still powerful indicators of (un)grammaticality: if we have never seen a PRN that starts with "has," or a span that begins with a quotation mark and ends with a close bracket, then we would like the model to be able to place negative weights on these features. Thus, we use a simple feature hashing scheme where positive features are indexed individually, while nega-

Features	Section	F1
RULE	4	73.0
+ SPAN FIRST WORD + SPAN LAST WORD + LENGTH	4.1	85.0
+ WORD BEFORE SPAN + WORD AFTER SPAN	4.2	89.0
+ WORD BEFORE SPLIT + WORD AFTER SPLIT	4.3	89.7
+ SPAN SHAPE	4.4	89.9

Table 1: Results for the Penn Treebank development set, reported in F1 on sentences of length ≤ 40 on Section 22, for a number of incrementally growing feature sets. We show that each feature type presented in Section 4 adds benefit over the previous, and in combination they produce a reasonably good yet simple parser.

tive features are bucketed together. During training there are no collisions between positive features, which generally receive positive weight, and negative features, which generally receive negative weight; only negative features can collide. Early experiments indicated that using a number of negative buckets equal to the number of positive features was effective.

4 Features

Our goal is to use surface features to replicate the functionality of other annotations, without increasing the state space of our grammar, meaning that the rules $\mathrm{rule}(r)$ remain simple, as does the state space used during inference.

Before we present our main features, we briefly discuss the issue of feature sparsity. While lexical features are a powerful driver of our parser, firing features on rare words would allow it to overfit the training data quite heavily. To that end, for the purposes of computing our features, a word is represented by its longest suffix that occurs 100 or more times in the training data (which will be the entire word, for common words).[1]

Table 1 shows the results of incrementally building up our feature set on the Penn Treebank development set. RULE specifies that we use only indicators on rule identity for binary production and nonterminal unaries. For this experiment and all others, we include a basic set of lexicon features, i.e. features on preterminal part-of-speech tags. A given preterminal unary at position i in the sentence includes features on the words (suffixes) at position $i-1$, i, and $i+1$. Because the lexicon is especially sensitive to morphological effects, we also fire features on all prefixes and suffixes of the current word up to length 5, regardless of frequency.

Subsequent lines in Table 1 indicate additional surface feature templates computed over the span, which are then conjoined with the rule identity as shown in Figure 1 to give additional features. In the rest of the section, we describe the features of this type that we use. Note that many of these features have been used before (Taskar et al., 2004; Finkel et al., 2008; Petrov and Klein, 2008b); our goal here is not to amass as many feature templates as possible, but rather to examine the extent to which a simple set of features can replace a complicated state space.

4.1 Basic Span Features

We start with some of the most obvious properties available to us, namely, the identity of the first and last words of a span. Because heads of constituents are often at the beginning or the end of a span, these feature templates can (noisily) capture monolexical properties of heads without having to incur the inferential cost of lexicalized annotations. For example, in English, the syntactic head of a verb phrase is typically at the beginning of the span, while the head of a simple noun phrase is the last word. Other languages, like Korean or Japanese, are more consistently head final.

Structural contexts like those captured by parent annotation (Johnson, 1998) are more subtle. Parent annotation can capture, for instance, the difference in distribution in NPs that have S as a parent (that is, subjects) and NPs under VPs (objects). We try to capture some of this same intuition by introducing a feature on the length of a span. For instance, VPs embedded in NPs tend to be short, usually as embedded gerund phrases. Because constituents in the treebank can be quite long, we bin our length features into 8 buckets, of

[1] Experiments with the Brown clusters (Brown et al., 1992) provided by Turian et al. (2010) in lieu of suffixes were not promising. Moreover, lowering this threshold did not improve performance.

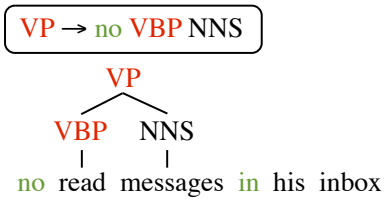


Figure 2: An example showing the utility of span context. The ambiguity about whether *read* is an adjective or a verb is resolved when we construct a VP and notice that the word proceeding it is unlikely.

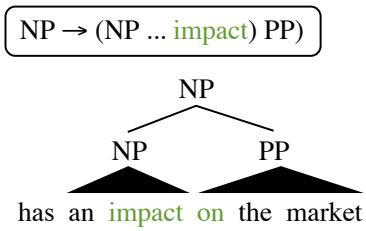


Figure 3: An example showing split point features disambiguating a PP attachment. Because *impact* is likely to take a PP, the monolexical indicator feature that conjoins *impact* with the appropriate rule will help us parse this example correctly.

lengths 1, 2, 3, 4, 5, 10, 20, and ≥21 words.

Adding these simple features (first word, last word, and lengths) as span features of the X-bar grammar already gives us a substantial improvement over our baseline system, improving the parser's performance from 73.0 F1 to 85.0 F1 (see Table 1).

4.2 Span Context Features

Of course, there is no reason why we should confine ourselves to just the words within the span: words outside the span also provide a rich source of context. As an example, consider disambiguating the POS tag of the word *read* in Figure 2. A VP is most frequently preceded by a subject NP, whose rightmost word is often its head. Therefore, we fire features that (separately) look at the words immediately preceding and immediately following the span.

4.3 Split Point Features

Another important source of features are the words at and around the split point of a binary rule application. Figure 3 shows an example of one instance of this feature template. *impact* is a noun that is more likely to take a PP than other nouns, and so we expect this feature to have high weight and encourage the attachment; this feature proves generally useful in resolving such cases of right-attachments to noun phrases, since the last word of the noun phrase is often the head. As another example, coordination can be represented by an indicator of the conjunction, which comes immediately after the split point. Finally, control structures with infinitival complements can be captured with a rule S → NP VP with the word "to" at the split point.

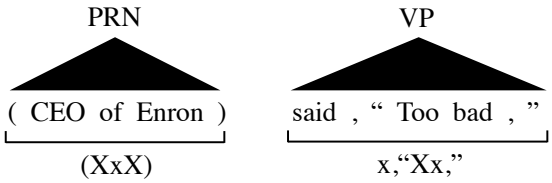


Figure 4: Computation of span shape features on two examples. Parentheticals, quotes, and other punctuation-heavy, short constituents benefit from being explicitly modeled by a descriptor like this.

4.4 Span Shape Features

We add one final feature characterizing the span, which we call span shape. Figure 4 shows how this feature is computed. For each word in the span,[2] we indicate whether that word begins with a capital letter, lowercase letter, digit, or punctuation mark. If it begins with punctuation, we indicate the punctuation mark explicitly. Figure 4 shows that this is especially useful in characterizing constructions such as parentheticals and quoted expressions. Because this feature indicates capitalization, it can also capture properties of NP internal structure relevant to named entities, and its sensitivity to capitalization and punctuation makes it useful for recognizing appositive constructions.

5 Annotations

We have built up a strong set of features by this point, but have not yet answered the question of whether or not grammar annotation is useful on top of them. In this section, we examine two of the most commonly used types of additional annotation, structural annotation, and lexical annotation.

[2]For longer spans, we only use words sufficiently close to the span's beginning and end.

Annotation	Dev, len ≤ 40
$v = 0, h = 0$	90.1
$v = 1, h = 0$	90.5
$v = 0, h = 1$	90.2
$v = 1, h = 1$	90.9
Lexicalized	90.3

Table 2: Results for the Penn Treebank development set, sentences of length ≤ 40, for different annotation schemes implemented on top of the X-bar grammar.

Recall from Section 3 that every span feature is conjoined with indicators over rules and rule parents to produce features over anchored rule productions; when we consider adding an annotation layer to the grammar, what that does is refine the rule indicators that are conjoined with every span feature. While this is a powerful way of refining features, we show that common successful annotation schemes provide at best modest benefit on top of the base parser.

5.1 Structural Annotation

The most basic, well-understood kind of annotation on top of an X-bar grammar is structural annotation, which annotates each nonterminal with properties of its environment (Johnson, 1998; Klein and Manning, 2003). This includes vertical annotation (parent, grandparent, etc.) as well as horizontal annotation (only partially Markovizing rules as opposed to using an X-bar grammar).

Table 2 shows the performance of our feature set in grammars with several different levels of structural annotation.[3] Klein and Manning (2003) find large gains (6% absolute improvement, 20% relative improvement) going from $v = 0, h = 0$ to $v = 1, h = 1$; however, we do not find the same level of benefit. To the extent that our parser needs to make use of extra information in order to apply a rule correctly, simply inspecting the input to determine this information appears to be almost as effective as relying on information threaded through the parser.

In Section 6 and Section 7, we use $v = 1$ and $h = 0$; we find that $v = 1$ provides a small, reliable improvement across a range of languages and tasks, whereas other annotations are less clearly beneficial.

[3]We use $v = 0$ to indicate no annotation, diverging from the notation in Klein and Manning (2003).

	Test ≤ 40	Test all
Berkeley	90.6	90.1
This work	89.9	89.2

Table 3: Final Parseval results for the $v = 1, h = 0$ parser on Section 23 of the Penn Treebank.

5.2 Lexical Annotation

Another commonly-used kind of structural annotation is lexicalization (Eisner, 1996; Collins, 1997; Charniak, 1997). By annotating grammar nonterminals with their headwords, the idea is to better model phenomena that depend heavily on the semantics of the words involved, such as coordination and PP attachment.

Table 2 shows results from lexicalizing the X-bar grammar; it provides meager improvements. One probable reason for this is that our parser already includes monolexical features that inspect the first and last words of each span, which captures the syntactic or the semantic head in many cases or can otherwise provide information about what the constituent's type may be and how it is likely to combine. Lexicalization allows us to capture bilexical relationships along dependency arcs, but it has been previously shown that these add only marginal benefit to Collins's model anyway (Gildea, 2001).

5.3 English Evaluation

Finally, Table 3 shows our final evaluation on Section 23 of the Penn Treebank. We use the $v = 1, h = 0$ grammar. While we do not do as well as the Berkeley parser, we will see in Section 6 that our parser does a substantially better job of generalizing to other languages.

6 Other Languages

Historically, many annotation schemes for parsers have required language-specific engineering: for example, lexicalized parsers require a set of head rules and manually-annotated grammars require detailed analysis of the treebank itself (Klein and Manning, 2003). A key strength of a parser that does not rely heavily on an annotated grammar is that it may be more portable to other languages. We show that this is indeed the case: on nine languages, our system is competitive with or better than the Berkeley parser, which is the best single

	Arabic	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish	Avg
	Dev, all lengths									
Berkeley	78.24	69.17	79.74	81.74	87.83	83.90	70.97	84.11	74.50	78.91
Berkeley-Rep	78.70	84.33	79.68	82.74	89.55	89.08	82.84	87.12	75.52	83.28
Our work	78.89	83.74	79.40	83.28	88.06	87.44	81.85	91.10	75.95	83.30
	Test, all lengths									
Berkeley	79.19	70.50	80.38	78.30	86.96	81.62	71.42	79.23	79.18	78.53
Berkeley-Tags	78.66	74.74	79.76	78.28	85.42	85.22	78.56	86.75	80.64	80.89
Our work	78.75	83.39	79.70	78.43	87.18	88.25	80.18	90.66	82.00	83.17

Table 4: Results for the nine treebanks in the SPMRL 2013 Shared Task; all values are F-scores for sentences of all lengths using the version of `evalb` distributed with the shared task. Berkeley-Rep is the best single parser from (Björkelund et al., 2013); we only compare to this parser on the development set because neither the system nor test set values are publicly available. Berkeley-Tags is a version of the Berkeley parser run by the task organizers where tags are provided to the model, and is the best single parser submitted to the official task. In both cases, we match or outperform the baseline parsers in aggregate and on the majority of individual languages.

parser[4] for the majority of cases we consider.

We evaluate on the constituency treebanks from the Statistical Parsing of Morphologically Rich Languages Shared Task (Seddah et al., 2013). We compare to the Berkeley parser (Petrov and Klein, 2007) as well as two variants. First, we use the "Replaced" system of Björkelund et al. (2013) (Berkeley-Rep), which is their best single parser.[5] The "Replaced" system modifies the Berkeley parser by replacing rare words with morphological descriptors of those words computed using language-specific modules, which have been hand-crafted for individual languages or are trained with additional annotation layers in the treebanks that we do not exploit. Unfortunately, Björkelund et al. (2013) only report results on the development set for the Berkeley-Rep model; however, the task organizers also use a version of the Berkeley parser provided with parts of speech from high-quality POS taggers for each language (Berkeley-Tags). These part-of-speech taggers often incorporate substantial knowledge of each language's morphology. Both Berkeley-Rep and Berkeley-Tags make up for some shortcomings of the Berkeley parser's unknown word model, which is tuned to English.

In Table 4, we see that our performance is overall substantially higher than that of the Berkeley parser. On the development set, we outperform the Berkeley parser and match the performance of the Berkeley-Rep parser. On the test set, we outperform both the Berkeley parser and the Berkeley-Tags parser on seven of nine languages, losing only on Arabic and French.

These results suggest that the Berkeley parser may be heavily fit to English, particularly in its lexicon. However, even when language-specific unknown word handling is added to the parser, our model still outperforms the Berkeley parser overall, showing that our model generalizes even better across languages than a parser for which this is touted as a strength (Petrov and Klein, 2007). Our span features appear to work well on both head-initial and head-final languages (see Basque and Korean in the table), and the fact that our parser performs well on such morphologically-rich languages as Hungarian indicates that our suffix model is sufficient to capture most of the morphological effects relevant to parsing. Of course, a language that was heavily prefixing would likely require this feature to be modified. Likewise, our parser does not perform as well on Arabic and Hebrew. These closely related languages use templatic morphology, for which suffixing is not appropriate; however, using additional surface features based on the output of a morphological analyzer did not lead to increased performance.

Finally, our high performance on languages such as Polish and Swedish, whose training treebanks consist of 6578 and 5000 sentences, respectively, show that our feature-rich model performs robustly even on treebanks much smaller than the Penn Treebank.[6]

[4] I.e. it does not use a reranking step or post-hoc combination of parser results.

[5] Their best parser, and the best overall parser from the shared task, is a reranked product of "Replaced" Berkeley parsers.

[6] The especially strong performance on Polish relative to other systems is partially a result of our model being able to produce unary chains of length two, which occur frequently in the Polish treebank (Björkelund et al., 2013).

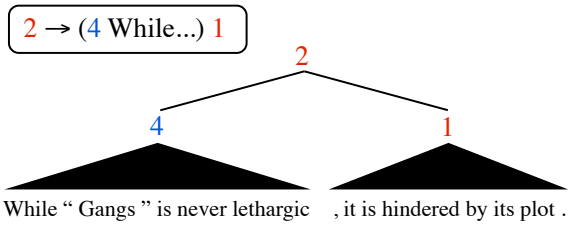


Figure 5: An example of a sentence from the Stanford Sentiment Treebank which shows the utility of our span features for this task. The presence of "While" under this kind of rule tells us that the sentiment of the constituent to the right dominates the sentiment to the left.

7 Sentiment Analysis

Finally, because the system is, at its core, a classifier of spans, it can be used equally well for tasks that do not normally use parsing algorithms. One example is sentiment analysis. While approaches to sentiment analysis often simply classify the sentence monolithically, treating it as a bag of n-grams (Pang et al., 2002; Pang and Lee, 2005; Wang and Manning, 2012), the recent dataset of Socher et al. (2013) imposes a layer of structure on the problem that we can exploit. They annotate every constituent in a number of training trees with an integer sentiment value from 1 (very negative) to 5 (very positive), opening the door for models such as ours to learn how syntax can structurally affect sentiment.[7]

Figure 5 shows an example that requires some analysis of sentence structure to correctly understand. The first constituent conveys positive sentiment with *never lethargic* and the second conveys negative sentiment with *hindered*, but to determine the overall sentiment of the sentence, we need to exploit the fact that *while* signals a discounting of the information that follows it. The grammar rule $2 \rightarrow 4\ 1$ already encodes the notion of the sentiment of the right child being dominant, so when this is conjoined with our span feature on the first word (*While*), we end up with a feature that captures this effect. Our features can also lexicalize on other discourse connectives such as *but* or *however*, which often occur at the split point between two spans.

[7]Note that the tree structure is assumed to be given; the problem is one of labeling a fixed parse backbone.

7.1 Adapting to Sentiment

Our parser is almost entirely unchanged from the parser that we used for syntactic analysis. Though the treebank grammar is substantially different, with the nonterminals consisting of five integers with very different semantics from syntactic nonterminals, we still find that parent annotation is effective and otherwise additional annotation layers are not useful.

One structural difference between sentiment analysis and syntactic parsing lies in where the relevant information is present in a span. Syntax is often driven by heads of constituents, which tend to be located at the beginning or the end, whereas sentiment is more likely to depend on modifiers such as adjectives, which are typically present in the middle of spans. Therefore, we augment our existing model with standard sentiment analysis features that look at unigrams and bigrams in the span (Wang and Manning, 2012). Moreover, the Stanford Sentiment Treebank is unique in that each constituent was annotated in isolation, meaning that context never affects sentiment and that every word always has the same tag. We exploit this by adding an additional feature template similar to our span shape feature from Section 4.4 which uses the (deterministic) tag for each word as its descriptor.

7.2 Results

We evaluated our model on the fine-grained sentiment analysis task presented in Socher et al. (2013) and compare to their released system. The task is to predict the root sentiment label of each parse tree; however, because the data is annotated with sentiment at each span of each parse tree, we can also evaluate how well our model does at these intermediate computations. Following their experimental conditions, we filter the test set so that it only contains trees with non-neutral sentiment labels at the root.

Table 5 shows that our model outperforms the model of Socher et al. (2013)—both the published numbers and latest released version—on the task of root classification, even though the system was not explicitly designed for this task. Their model has high capacity to model complex interactions of words through a combinatory tensor, but it appears that our simpler, feature-driven model is just as effective at capturing the key effects of compositionality for sentiment analysis.

	Root	All Spans
Non-neutral Dev (872 trees)		
Stanford CoreNLP current	50.7	80.8
This work	53.1	80.5
Non-neutral Test (1821 trees)		
Stanford CoreNLP current	49.1	80.2
Stanford EMNLP 2013	45.7	80.7
This work	49.6	80.4

Table 5: Fine-grained sentiment analysis results on the Stanford Sentiment Treebank of Socher et al. (2013). We compare against the printed numbers in Socher et al. (2013) as well as the performance of the corresponding release, namely the sentiment component in the latest version of the Stanford CoreNLP at the time of this writing. Our model handily outperforms the results from Socher et al. (2013) at root classification and edges out the performance of the latest version of the Stanford system. On all spans of the tree, our model has comparable accuracy to the others.

8 Conclusion

To date, the most successful constituency parsers have largely been generative, and operate by refining the grammar either manually or automatically so that relevant information is available locally to each parsing decision. Our main contribution is to show that there is an alternative to such annotation schemes: namely, conditioning on the input and firing features based on anchored spans. We build up a small set of feature templates as part of a discriminative constituency parser and outperform the Berkeley parser on a wide range of languages. Moreover, we show that our parser is adaptable to other tree-structured tasks such as sentiment analysis; we outperform the recent system of Socher et al. (2013) and obtain state of the art performance on their dataset.

Our system is available as open-source at `https://www.github.com/dlwh/epic`.

Acknowledgments

This work was partially supported by BBN under DARPA contract HR0011-12-C-0014, by a Google PhD fellowship to the first author, and an NSF fellowship to the second. We further gratefully acknowledge a hardware donation by NVIDIA Corporation.

References

Anders Björkelund, Ozlem Cetinoglu, Richárd Farkas, Thomas Mueller, and Wolfgang Seeker. 2013. (Re)ranking Meets Morphosyntax: State-of-the-art Results from the SPMRL 2013 Shared Task. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*.

Rens Bod. 1993. Using an Annotated Corpus As a Stochastic Grammar. In *Proceedings of the Sixth Conference on European Chapter of the Association for Computational Linguistics*.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine N-best Parsing and MaxEnt Discriminative Reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*.

Eugene Charniak. 1997. Statistical Techniques for Natural Language Parsing. *AI Magazine*, 18:33–44.

Michael Collins and Terry Koo. 2005. Discriminative Reranking for Natural Language Parsing. *Computational Linguistics*, 31(1):25–70, March.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *ACL*, pages 16–23.

John Duchi, Elad Hazan, and Yoram Singer. 2010. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *COLT*.

Jason Eisner. 1996. Three New Probabilistic Models for Dependency Parsing: An Exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*.

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *ACL 2008*, pages 959–967.

Daniel Gildea. 2001. Corpus variation and parser performance. In *Proceedings of Empirical Methods in Natural Language Processing*.

David Hall and Dan Klein. 2012. Training factored PCFGs with expectation propagation. In *EMNLP*.

James Henderson. 2003. Inducing History Representations for Broad Coverage Statistical Parsing. In *Proceedings of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*, pages 586–594, Columbus, Ohio, June. Association for Computational Linguistics.

Mark Johnson. 1998. PCFG Models of Linguistic Tree Representations. *Computational Linguistics*, 24(4):613–632, December.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *ACL*, pages 423–430.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning*.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *ACL*, pages 75–82, Morristown, NJ, USA.

Bo Pang and Lillian Lee. 2005. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs Up?: Sentiment Classification Using Machine Learning Techniques. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*.

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

Slav Petrov and Dan Klein. 2008a. Discriminative log-linear grammars with latent variables. In *NIPS*, pages 1153–1160.

Slav Petrov and Dan Klein. 2008b. Sparse multi-scale grammars for discriminative latent variable parsing. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 867–876, Honolulu, Hawaii, October. Association for Computational Linguistics.

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

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, and Alina Wróblewska. 2013. Overview of the SPMRL 2013 Shared Task: A Cross-Framework Evaluation of Parsing Morphologically Rich Languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*.

Khalil Sima'an. 2000. Tree-gram Parsing Lexical Dependencies and Structural Relations. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of Empirical Methods in Natural Language Processing*.

Ben Taskar, Dan Klein, Michael Collins, Daphne Koller, and Christopher Manning. 2004. Max-Margin Parsing. In *In Proceedings of Empirical Methods in Natural Language Processing*.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich Part-of-speech Tagging with a Cyclic Dependency Network. In *Proceedings of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics.

Sida Wang and Christopher Manning. 2012. Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors

Marco Baroni and **Georgiana Dinu** and **Germán Kruszewski**
Center for Mind/Brain Sciences (University of Trento, Italy)
(marco.baroni|georgiana.dinu|german.kruszewski)@unitn.it

Abstract

Context-predicting models (more commonly known as embeddings or neural language models) are the new kids on the distributional semantics block. Despite the buzz surrounding these models, the literature is still lacking a systematic comparison of the predictive models with classic, count-vector-based distributional semantic approaches. In this paper, we perform such an extensive evaluation, on a wide range of lexical semantics tasks and across many parameter settings. The results, to our own surprise, show that the buzz is fully justified, as the context-predicting models obtain a thorough and resounding victory against their count-based counterparts.

1 Introduction

A long tradition in computational linguistics has shown that contextual information provides a good approximation to word meaning, since semantically similar words tend to have similar contextual distributions (Miller and Charles, 1991). In concrete, *distributional semantic models* (DSMs) use vectors that keep track of the contexts (e.g., co-occurring words) in which target terms appear in a large corpus as proxies for meaning representations, and apply geometric techniques to these vectors to measure the similarity in meaning of the corresponding words (Clark, 2013; Erk, 2012; Turney and Pantel, 2010).

It has been clear for decades now that raw co-occurrence counts don't work that well, and DSMs achieve much higher performance when various transformations are applied to the raw vectors, for example by reweighting the counts for context informativeness and smoothing them with dimensionality reduction techniques. This vector optimization process is generally unsupervised, and based on independent considerations (for example, context reweighting is often justified by information-theoretic considerations, dimensionality reduction optimizes the amount of preserved variance, etc.). Occasionally, some kind of indirect supervision is used: Several parameter settings are tried, and the best setting is chosen based on performance on a semantic task that has been selected for tuning.

The last few years have seen the development of a new generation of DSMs that frame the vector estimation problem directly as a supervised task, where the weights in a word vector are set to maximize the probability of the contexts in which the word is observed in the corpus (Bengio et al., 2003; Collobert and Weston, 2008; Collobert et al., 2011; Huang et al., 2012; Mikolov et al., 2013a; Turian et al., 2010). The traditional construction of context vectors is turned on its head: Instead of first collecting context vectors and then reweighting these vectors based on various criteria, the vector weights are directly set to optimally predict the contexts in which the corresponding words tend to appear. Since similar words occur in similar contexts, the system naturally learns to assign similar vectors to similar words.

This new way to train DSMs is attractive because it replaces the essentially heuristic stacking of vector transforms in earlier models with a single, well-defined supervised learning step. At the same time, supervision comes at no manual annotation cost, given that the context windows used for training can be automatically extracted from an unannotated corpus (indeed, they are the very same data used to build traditional DSMs). Moreover, at least some of the relevant methods can efficiently scale up to process very large amounts of input data.[1]

[1]The idea to directly learn a parameter vector based on an objective optimum function is shared by Latent Dirichlet

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 238–247,
Baltimore, Maryland, USA, June 23-25 2014.

We will refer to DSMs built in the traditional way as *count* models (since they initialize vectors with co-occurrence counts), and to their training-based alternative as *predict(ive)* models.[2] Now, the most natural question to ask, of course, is which of the two approaches is best in empirical terms. Surprisingly, despite the long tradition of extensive evaluations of alternative count DSMs on standard benchmarks (Agirre et al., 2009; Baroni and Lenci, 2010; Bullinaria and Levy, 2007; Bullinaria and Levy, 2012; Sahlgren, 2006; Padó and Lapata, 2007), the existing literature contains very little in terms of direct comparison of count vs. predictive DSMs. This is in part due to the fact that context-predicting vectors were first developed as an approach to language modeling and/or as a way to initialize feature vectors in neural-network-based "deep learning" NLP architectures, so their effectiveness as semantic representations was initially seen as little more than an interesting side effect. Sociological reasons might also be partly responsible for the lack of systematic comparisons: Context-predictive models were developed within the neural-network community, with little or no awareness of recent DSM work in computational linguistics.

Whatever the reasons, we know of just three works reporting direct comparisons, all limited in their scope. Huang et al. (2012) compare, in passing, one count model and several predict DSMs on the standard WordSim353 benchmark (Table 3 of their paper). In this experiment, the count model actually outperforms the best predictive approach. Instead, in a word-similarity-in-context task (Table 5), the best predict model outperforms the count model, albeit not by a large margin.

Blacoe and Lapata (2012) compare count and predict representations as input to composition functions. Count vectors make for better inputs in a phrase similarity task, whereas the two representations are comparable in a paraphrase classification experiment.[3]

Finally, Mikolov et al. (2013d) compare their predict models to "Latent Semantic Analysis" (LSA) count vectors on syntactic and semantic analogy tasks, finding that the predict models are highly superior. However, they provide very little details about the LSA count vectors they use.[4]

In this paper, we overcome the comparison scarcity problem by providing a direct evaluation of count and predict DSMs across many parameter settings and on a large variety of mostly standard lexical semantics benchmarks. Our title already gave away what we discovered.

2 Distributional semantic models

Both count and predict models are extracted from a corpus of about 2.8 billion tokens constructed by concatenating ukWaC,[5] the English Wikipedia[6] and the British National Corpus.[7] For both model types, we consider the top 300K most frequent words in the corpus both as target and context elements.

2.1 Count models

We prepared the count models using the DISSECT toolkit.[8] We extracted count vectors from symmetric context windows of two and five words to either side of target. We considered two weighting schemes: positive Pointwise Mutual Information and Local Mutual Information (akin to the widely used Log-Likelihood Ratio scheme) (Evert, 2005). We used both full and compressed vectors. The latter were obtained by applying the Singular Value Decomposition (Golub and Van Loan, 1996) or Non-negative Matrix Factorization (Lee and Seung, 2000), Lin (2007) algorithm, with reduced sizes ranging from 200 to 500 in steps of 100. In total, 36 count models were evaluated.

Count models have such a long and rich history that we can only explore a small subset of the counting, weighting and compressing methods proposed in the literature. However, it is worth pointing out that the evaluated parameter subset encompasses settings (narrow context window, positive PMI, SVD reduction) that have been

Allocation (LDA) models (Blei et al., 2003; Griffiths et al., 2007), where parameters are set to optimize the joint probability distribution of words and documents. However, the fully probabilistic LDA models have problems scaling up to large data sets.

[2] We owe the first term to Hinrich Schütze (p.c.). Predictive DSMs are also called neural language models, because their supervised context prediction training is performed with neural networks, or, more cryptically, "embeddings".

[3] We refer here to the updated results reported in the erratum at `http://homepages.inf.ed.ac.uk/s1066731/pdf/emnlp2012erratum.pdf`

[4] Chen et al. (2013) present an extended empirical evaluation, that is however limited to alternative context-predictive models, and does not include the word2vec variant we use here.

[5] `http://wacky.sslmit.unibo.it`

[6] `http://en.wikipedia.org`

[7] `http://www.natcorp.ox.ac.uk`

[8] `http://clic.cimec.unitn.it/composes/toolkit/`

found to be most effective in the systematic explorations of the parameter space conducted by Bullinaria and Levy (2007; 2012).

2.2 Predict models

We trained our predict models with the word2vec toolkit.[9] The toolkit implements both the skip-gram and CBOW approaches of Mikolov et al. (2013a; 2013c). We experimented only with the latter, which is also the more computationally-efficient model of the two, following Mikolov et al. (2013b) which recommends CBOW as more suitable for larger datasets.

The CBOW model learns to predict the word in the middle of a symmetric window based on the sum of the vector representations of the words in the window. We considered context windows of 2 and 5 words to either side of the central element. We vary vector dimensionality within the 200 to 500 range in steps of 100. The word2vec toolkit implements two efficient alternatives to the standard computation of the output word probability distributions by a softmax classifier. Hierarchical softmax is a computationally efficient way to estimate the overall probability distribution using an output layer that is proportional to $log(unigram.perplexity(W))$ instead of W (for W the vocabulary size). As an alternative, negative sampling estimates the probability of an output word by learning to distinguish it from draws from a noise distribution. The number of these draws (number of *negative samples*) is given by a parameter k. We test both hierarchical softmax and negative sampling with k values of 5 and 10. Very frequent words such as *the* or *a* are not very informative as context features. The word2vec toolkit implements a method to downsize their effect (and simultaneously improve speed performance). More precisely, words in the training data are discarded with a probability that is proportional to their frequency (capturing the same intuition that motivates traditional count vector weighting measures such as PMI). This is controlled by a parameter t and words that occur with higher frequency than t are aggressively subsampled. We train models without subsampling and with subsampling at $t = 1e^{-5}$ (the toolkit page suggests $1e^{-3} - 1e^{-5}$ as a useful range based on empirical observations).

In total, we evaluate 48 predict models, a number comparable to that of the count models we consider.

2.3 Out-of-the-box models

Baroni and Lenci (2010) make the vectors of their best-performing *Distributional Memory* (dm) model available.[10] This model, based on the same input corpus we use, exemplifies a "linguistically rich" count-based DSM, that relies on lemmas instead or raw word forms, and has dimensions that encode the syntactic relations and/or lexico-syntactic patterns linking targets and contexts. Baroni and Lenci showed, in a large scale evaluation, that dm reaches near-state-of-the-art performance in a variety of semantic tasks.

We also experiment with the popular predict vectors made available by Ronan Collobert.[11] Following the earlier literature, with refer to them as *Collobert and Weston* (cw) vectors. These are 100-dimensional vectors trained for two months (!) on the Wikipedia. In particular, the vectors were trained to optimize the task of choosing the right word over a random alternative in the middle of an 11-word context window (Collobert et al., 2011).

3 Evaluation materials

We test our models on a variety of benchmarks, most of them already widely used to test and compare DSMs. The following benchmark descriptions also explain the figures of merit and state-of-the-art results reported in Table 2.

Semantic relatedness A first set of semantic benchmarks was constructed by asking human subjects to rate the degree of semantic similarity or relatedness between two words on a numerical scale. The performance of a computational model is assessed in terms of correlation between the average scores that subjects assigned to the pairs and the cosines between the corresponding vectors in the model space (following the previous art, we use Pearson correlation for rg, Spearman in all other cases). The classic data set of Rubenstein and Goodenough (1965) (rg) consists of 65 noun pairs. State of the art performance on this set has been reported by Hassan and Mihalcea (2011) using a technique that exploits the Wikipedia linking structure and word sense disambiguation techniques. Finkelstein et al. (2002)

[9] https://code.google.com/p/word2vec/

[10] http://clic.cimec.unitn.it/dm/

[11] http://ronan.collobert.com/senna/

introduced the widely used WordSim353 set (ws) that, as the name suggests, consists of 353 pairs. The current state of the art is reached by Halawi et al. (2012) with a method that is in the spirit of the predict models, but lets synonymy information from WordNet constrain the learning process (by favoring solutions in which WordNet synonyms are near in semantic space). Agirre et al. (2009) split the ws set into similarity (wss) and relatedness (wsr) subsets. The first contains tighter taxonomic relations, such as synonymy and cohyponymy (*king/queen*) whereas the second encompasses broader, possibly topical or syntagmatic relations (*family/planning*). We report state-of-the-art performance on the two subsets from the work of Agirre and colleagues, who used different kinds of count vectors extracted from a very large corpus (orders of magnitude larger than ours). Finally, we use (the test section of) MEN (men), that comprises 1,000 word pairs. Bruni et al. (2013), the developers of this benchmark, achieve state-of-the-art performance by extensive tuning on *ad-hoc* training data, and by using both textual and image-extracted features to represent word meaning.

Synonym detection The classic TOEFL (toefl) set was introduced by Landauer and Dumais (1997). It contains 80 multiple-choice questions that pair a target term with 4 synonym candidates. For example, for the target *levied* one must choose between *imposed* (correct), *believed*, *requested* and *correlated*. The DSMs compute cosines of each candidate vector with the target, and pick the candidate with largest cosine as their answer. Performance is evaluated in terms of correct-answer accuracy. Bullinaria and Levy (2012) achieved 100% accuracy by a very thorough exploration of the count model parameter space.

Concept categorization Given a set of nominal concepts, the task is to group them into natural categories (e.g., *helicopters* and *motorcycles* should go to the *vehicle* class, *dogs* and *elephants* into the *mammal* class). Following previous art, we tackle categorization as an unsupervised clustering task. The vectors produced by a model are clustered into n groups (with n determined by the gold standard partition) using the CLUTO toolkit (Karypis, 2003), with the repeated bisections with global optimization method and CLUTO's default settings otherwise (these are standard choices in the literature). Performance is evaluated in terms of *purity*, a measure of the extent to which each cluster contains concepts from a single gold category. If the gold partition is reproduced perfectly, purity reaches 100%; it approaches 0 as cluster quality deteriorates. The Almuhareb-Poesio (ap) benchmark contains 402 concepts organized into 21 categories (Almuhareb, 2006). State-of-the-art purity was reached by Rothenhäusler and Schütze (2009) with a count model based on carefully crafted syntactic links. The ESSLLI 2008 Distributional Semantic Workshop shared-task set (esslli) contains 44 concepts to be clustered into 6 categories (Baroni et al., 2008) (we ignore here the 3- and 2-way higher-level partitions coming with this set). Katrenko and Adriaans (2008) reached top performance on this set using the full Web as a corpus and manually crafted, linguistically motivated patterns. Finally, the Battig (battig) test set introduced by Baroni et al. (2010) includes 83 concepts from 10 categories. Current state of the art was reached by the window-based count model of Baroni and Lenci (2010).

Selectional preferences We experiment with two data sets that contain verb-noun pairs that were rated by subjects for the typicality of the noun as a subject or object of the verb (e.g., *people* received a high average score as subject of *to eat*, and a low score as object of the same verb). We follow the procedure proposed by Baroni and Lenci (2010) to tackle this challenge: For each verb, we use the corpus-based tuples they make available to select the 20 nouns that are most strongly associated to the verb as subjects or objects, and we average the vectors of these nouns to obtain a "prototype" vector for the relevant argument slot. We then measure the cosine of the vector for a target noun with the relevant prototype vector (e.g., the cosine of *people* with the *eating* subject prototype vector). Systems are evaluated by Spearman correlation of these cosines with the averaged human typicality ratings. Our first data set was introduced by Ulrike Padó (2007) and includes 211 pairs (up). Top-performance was reached by the supervised count vector system of Herdağdelen and Baroni (2009) (supervised in the sense that they directly trained a classifier on gold data, as opposed to the 0-cost supervision of the context-learning methods). The mcrae set (McRae et al., 1998) consists of 100 noun–verb pairs, with top performance reached by the DepDM system of Baroni and Lenci (2010), a count DSM relying on

syntactic information.

Analogy While all the previous data sets are relatively standard in the DSM field to test traditional count models, our last benchmark was introduced in Mikolov et al. (2013a) specifically to test predict models. The data-set contains about 9K semantic and 10.5K syntactic analogy questions. A semantic question gives an example pair (*brother-sister*), a test word (*grandson*) and asks to find another word that instantiates the relation illustrated by the example with respect to the test word (*granddaughter*). A syntactic question is similar, but in this case the relationship is of a grammatical nature (*work–works*, *speak… speaks*). Mikolov and colleagues tackle the challenge by subtracting the second example term vector from the first, adding the test term, and looking for the nearest neighbour of the resulting vector (what is the nearest neighbour of $\vec{brother} - \vec{sister} + \vec{grandson}$?). Systems are evaluated in terms of proportion of questions where the nearest neighbour from the whole semantic space is the correct answer (the given example and test vector triples are excluded from the nearest neighbour search). Mikolov et al. (2013a) reach top accuracy on the syntactic subset (ansyn) with a CBOW predict model akin to ours (but trained on a corpus twice as large). Top accuracy on the entire data set (an) and on the semantic subset (ansem) was reached by Mikolov et al. (2013c) using a skip-gram predict model. Note however that, because of the way the task is framed, performance also depends on the size of the vocabulary to be searched: Mikolov et al. (2013a) pick the nearest neighbour among vectors for 1M words, Mikolov et al. (2013c) among 700K words, and we among 300K words.

Some characteristics of the benchmarks we use are summarized in Table 1.

4 Results

Table 2 summarizes the evaluation results. The first block of the table reports the maximum per-task performance (across all considered parameter settings) for count and predict vectors. The latter emerge as clear winners, with a large margin over count vectors in most tasks. Indeed, the predictive models achieve an impressive overall performance, beating the current state of the art in several cases, and approaching it in many more. It is worth stressing that, as reviewed in Section 3, the state-of-the-art results were obtained in almost all cases using specialized approaches that rely on external knowledge, manually-crafted rules, parsing, larger corpora and/or task-specific tuning. Our predict results were instead achieved by simply downloading the word2vec toolkit and running it with a range of parameter choices recommended by the toolkit developers.

The success of the predict models cannot be blamed on poor performance of the count models. Besides the fact that this would not explain the near-state-of-the-art performance of the predict vectors, the count model results are actually quite good in absolute terms. Indeed, in several cases they are close, or even better than those attained by dm, a linguistically-sophisticated count-based approach that was shown to reach top performance across a variety of tasks by Baroni and Lenci (2010).

Interestingly, count vectors achieve performance comparable to that of predict vectors only on the selectional preference tasks. The up task in particular is also the only benchmark on which predict models are seriously lagging behind state-of-the-art and dm performance. Recall from Section 3 that we tackle selectional preference by creating average vectors representing typical verb arguments. We conjecture that this averaging approach, that worked well for dm vectors, might be problematic for prediction-trained vectors, and we plan to explore alternative methods to build the prototypes in future research.

Are our results robust to parameter choices, or are they due to very specific and brittle settings? The next few blocks of Table 2 address this question. The second block reports results obtained with single count and predict models that are best in terms of average performance rank across tasks (these are the models on the top rows of tables 3 and 4, respectively). We see that, for both approaches, performance is not seriously affected by using the single best setup rather than task-specific settings, except for a considerable drop in performance for the best predict model on esslli (due to the small size of this data set?), and an even more dramatic drop of the count model on ansem. A more cogent and interesting evaluation is reported in the third block of Table 2, where we see what happens if we use the single models with *worst* performance across tasks (recall from Section 2 above that, in any case, we are exploring a space of reasonable parameter settings, of the sort that an

name	task	measure	source	soa
rg	relatedness	Pearson	Rubenstein and Goodenough (1965)	Hassan and Mihalcea (2011)
ws	relatedness	Spearman	Finkelstein et al. (2002)	Halawi et al. (2012)
wss	relatedness	Spearman	Agirre et al. (2009)	Agirre et al. (2009)
wsr	relatedness	Spearman	Agirre et al. (2009)	Agirre et al. (2009)
men	relatedness	Spearman	Bruni et al. (2013)	Bruni et al. (2013)
toefl	synonyms	accuracy	Landauer and Dumais (1997)	Bullinaria and Levy (2012)
ap	categorization	purity	Almuhareb (2006)	Rothenhäusler and Schütze (2009)
esslli	categorization	purity	Baroni et al. (2008)	Katrenko and Adriaans (2008)
battig	categorization	purity	Baroni et al. (2010)	Baroni and Lenci (2010)
up	sel pref	Spearman	Padó (2007)	Herdağdelen and Baroni (2009)
mcrae	sel pref	Spearman	McRae et al. (1998)	Baroni and Lenci (2010)
an	analogy	accuracy	Mikolov et al. (2013a)	Mikolov et al. (2013c)
ansyn	analogy	accuracy	Mikolov et al. (2013a)	Mikolov et al. (2013a)
ansem	analogy	accuracy	Mikolov et al. (2013a)	Mikolov et al. (2013c)

Table 1: Benchmarks used in experiments, with type of task, figure of merit (measure), original reference (source) and reference to current state-of-the-art system (soa).

	rg	ws	wss	wsr	men	toefl	ap	esslli	battig	up	mcrae	an	ansyn	ansem
best setup on each task														
cnt	74	62	70	59	72	76	66	84	98	41	27	49	43	60
pre	84	75	**80**	**70**	**80**	91	75	86	**99**	41	28	**68**	**71**	**66**
best setup across tasks														
cnt	70	62	70	57	72	76	64	84	98	37	27	43	41	44
pre	83	73	78	68	**80**	86	71	77	98	41	26	67	69	64
worst setup across tasks														
cnt	11	16	23	4	21	49	24	43	38	-6	-10	1	0	1
pre	74	60	73	48	68	71	65	82	88	33	20	27	40	10
best setup on rg														
cnt	(74)	59	66	52	71	64	64	84	98	37	20	35	42	26
pre	(84)	71	76	64	79	85	72	84	98	39	25	66	70	61
other models														
soa	**86**	**81**	77	62	76	**100**	**79**	**91**	96	**60**	**32**	61	64	61
dm	82	35	60	13	42	77	76	84	94	51	29	NA	NA	NA
cw	48	48	61	38	57	56	58	61	70	28	15	11	12	9

Table 2: Performance of count (cnt), predict (pre), dm and cw models on all tasks. See Section 3 and Table 1 for figures of merit and state-of-the-art results (soa). Since dm has very low coverage of the an* data sets, we do not report its performance there.

experimenter might be tempted to choose without tuning). The count model performance is severely affected by this unlucky choice (2-word window, Local Mutual Information, NMF, 400 dimensions, mean performance rank: 83), whereas the predict approach is much more robust: To put its worst instantiation (2-word window, hierarchical softmax, no subsampling, 200 dimensions, mean rank: 51) into perspective, its performance is more than 10% below the *best* count model only for the an and ansem tasks, and actually higher than it in 3 cases (note how on esslli the worst predict models performs much better than the best one, confirming our suspicion about the brittleness of this small data set). The fourth block reports performance in what might be the most realistic scenario, namely by tuning the parameters on a development task. Specifically, we pick the models that work best on the small rg set, and report their performance on all tasks (we obtained similar results by picking other tuning sets). The selected count model is the third best overall model of its class as reported in Table 3. The selected predict model is the fourth best model in Table 4. The overall count performance is not greatly affected by this choice. Again, predict models confirm their robustness, in that their rg-tuned performance is always close (and in 3 cases better) than the one achieved by the best overall setup.

Tables 3 and 4 let us take a closer look at the most important count and predict parameters, by reporting the characteristics of the best models (in terms of average performance-based ranking across tasks) from both classes. For the count models, PMI is clearly the better weighting scheme, and SVD outperforms NMF as a dimensionality reduction technique. However, no compression at all (using all 300K original dimensions) works best. Compare this to the best overall predict vectors, that have 400 dimensions only, making them much more practical to use. For the predict models, we observe in Table 4 that negative sampling, where the task is to distinguish the target output word from samples drawn from the noise distribution, outperforms the more costly hierarchical softmax method. Subsampling frequent words, which downsizes the importance of these words similarly to PMI weighting in count models, is also bringing significant improvements.

Finally, we go back to Table 2 to point out the poor performance of the out-of-the-box cw model.

window	weight	compress	dim.	mean rank
2	PMI	no	300K	35
5	PMI	no	300K	38
2	PMI	SVD	500	42
2	PMI	SVD	400	46
5	PMI	SVD	500	47
2	PMI	SVD	300	50
5	PMI	SVD	400	51
2	PMI	NMF	300	52
2	PMI	NMF	400	53
5	PMI	SVD	300	53

Table 3: Top count models in terms of mean performance-based model ranking across all tasks. The first row states that the window-2, PMI, 300K count model was the best count model, and, across all tasks, its average rank, when ALL models are decreasingly ordered by performance, was 35. See Section 2.1 for explanation of the parameters.

We must leave the investigation of the parameters that make our predict vectors so much better than cw (more varied training corpus? window size? objective function being used? subsampling? ...) to further work. Still, our results show that it's not just training by context prediction that ensures good performance. The cw approach is very popular (for example both Huang et al. (2012) and Blacoe and Lapata (2012) used it in the studies we discussed in Section 1). Had we also based our systematic comparison of count and predict vectors on the cw model, we would have reached opposite conclusions from the ones we can draw from our word2vec-trained vectors!

5 Conclusion

This paper has presented the first systematic comparative evaluation of count and predict vectors. As seasoned distributional semanticists with thorough experience in developing and using count vectors, we set out to conduct this study because we were annoyed by the triumphalist overtones often surrounding predict models, despite the almost complete lack of a proper comparison to count vectors.[12] Our secret wish was to discover that it is all hype, and count vectors are far superior to their predictive counterparts. A more realistic expec-

[12]Here is an example, where word2vec is called the crown jewel of natural language processing: `http://bit.ly/1ipv72M`

win.	hier. softm.	neg. samp.	subsamp.	dim	mean rank
5	no	10	yes	400	10
2	no	10	yes	300	13
5	no	5	yes	400	13
5	no	5	yes	300	13
5	no	10	yes	300	13
2	no	10	yes	400	13
2	no	5	yes	400	15
5	no	10	yes	200	15
2	no	10	yes	500	15
2	no	5	yes	300	16

Table 4: Top predict models in terms of mean performance-based model ranking across all tasks. See Section 2.2 for explanation of the parameters.

tation was that a complex picture would emerge, with predict and count vectors beating each other on different tasks. Instead, we found that the predict models are so good that, while the triumphalist overtones still sound excessive, there are very good reasons to switch to the new architecture. However, due to space limitations we have only focused here on quantitative measures: It remains to be seen whether the two types of models are complementary in the errors they make, in which case combined models could be an interesting avenue for further work.

The space of possible parameters of count DSMs is very large, and it's entirely possible that some options we did not consider would have improved count vector performance somewhat. Still, given that the predict vectors also outperformed the syntax-based dm model, and often approximated state-of-the-art performance, a more proficuous way forward might be to focus on parameters and extensions of the predict models instead: After all, we obtained our already excellent results by just trying a few variations of the word2vec defaults. Add to this that, beyond the standard lexical semantics challenges we tested here, predict models are currently been successfully applied in cutting-edge domains such as representing phrases (Mikolov et al., 2013c; Socher et al., 2012) or fusing language and vision in a common semantic space (Frome et al., 2013; Socher et al., 2013).

Based on the results reported here and the considerations we just made, we would certainly recommend anybody interested in using DSMs for theoretical or practical applications to go for the predict models, with the important caveat that they are not all created equal (cf. the big difference between word2vec and cw models). At the same time, given the large amount of work that has been carried out on count DSMs, we would like to explore, in the near future, how certain questions and methods that have been considered with respect to traditional DSMs will transfer to predict models. For example, the developers of Latent Semantic Analysis (Landauer and Dumais, 1997), Topic Models (Griffiths et al., 2007) and related DSMs have shown that the dimensions of these models can be interpreted as general "latent" semantic domains, which gives the corresponding models some *a priori* cognitive plausibility while paving the way for interesting applications. Another important line of DSM research concerns "context engineering": There has been for example much work on how to encode syntactic information into context features (Padó and Lapata, 2007), and more recent studies construct and combine feature spaces expressing topical vs. functional information (Turney, 2012). To give just one last example, distributional semanticists have looked at whether certain properties of vectors reflect semantic relations in the expected way: e.g., whether the vectors of hypernyms "distributionally include" the vectors of hyponyms in some mathematical precise sense.

Do the dimensions of predict models also encode latent semantic domains? Do these models afford the same flexibility of count vectors in capturing linguistically rich contexts? Does the structure of predict vectors mimic meaningful semantic relations? Does all of this even matter, or are we on the cusp of discovering radically new ways to tackle the same problems that have been approached as we just sketched in traditional distributional semantics?

Either way, the results of the present investigation indicate that these are important directions for future research in computational semantics.

Acknowledgments

We acknowledge ERC 2011 Starting Independent Research Grant n. 283554 (COMPOSES).

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Pasça, and Aitor Soroa. 2009.

A study on similarity and relatedness using distributional and WordNet-based approaches. In *Proceedings of HLT-NAACL*, pages 19–27, Boulder, CO.

Abdulrahman Almuhareb. 2006. *Attributes in Lexical Acquisition*. Phd thesis, University of Essex.

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

Marco Baroni, Stefan Evert, and Alessandro Lenci, editors. 2008. *Bridging the Gap between Semantic Theory and Computational Simulations: Proceedings of the ESSLLI Workshop on Distributional Lexical Semantic*. FOLLI, Hamburg.

Marco Baroni, Eduard Barbu, Brian Murphy, and Massimo Poesio. 2010. Strudel: A distributional semantic model based on properties and types. *Cognitive Science*, 34(2):222–254.

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

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of EMNLP*, pages 546–556, Jeju Island, Korea.

David Blei, Andrew Ng, and Michael Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Elia Bruni, Nam Khanh Tran, and Marco Baroni. 2013. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*. In press; `http://clic.cimec.unitn.it/marco/publications/mmds-jair.pdf`.

John Bullinaria and Joseph Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, 39:510–526.

John Bullinaria and Joseph Levy. 2012. Extracting semantic representations from word co-occurrence statistics: Stop-lists, stemming and SVD. *Behavior Research Methods*, 44:890–907.

Yanqing Chen, Bryan Perozzi, Rami Al-Rfou', and Steven Skiena. 2013. The expressive power of word embeddings. In *Proceedings of the ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, Atlanta, GA. Published online: `https://sites.google.com/site/deeplearningicml2013/accepted_papers`.

Stephen Clark. 2013. Vector space models of lexical meaning. In Shalom Lappin and Chris Fox, editors, *Handbook of Contemporary Semantics, 2nd ed.* Blackwell, Malden, MA. In press; `http://www.cl.cam.ac.uk/~sc609/pubs/sem_handbook.pdf`.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*, pages 160–167, Helsinki, Finland.

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

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

Stefan Evert. 2005. *The Statistics of Word Cooccurrences*. Ph.D dissertation, Stuttgart University.

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

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

Gene Golub and Charles Van Loan. 1996. *Matrix Computations (3rd ed.)*. JHU Press, Baltimore, MD.

Tom Griffiths, Mark Steyvers, and Josh Tenenbaum. 2007. Topics in semantic representation. *Psychological Review*, 114:211–244.

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. 2012. Large-scale learning of word relatedness with constraints. In *Proceedings of KDD*, pages 1406–1414.

Samer Hassan and Rada Mihalcea. 2011. Semantic relatedness using salient semantic analysis. In *Proceedings of AAAI*, pages 884–889, San Francisco, CA.

Amaç Herdağdelen and Marco Baroni. 2009. BagPack: A general framework to represent semantic relations. In *Proceedings of GEMS*, pages 33–40, Athens, Greece.

Eric Huang, Richard Socher, Christopher Manning, and Andrew Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of ACL*, pages 873–882, Jeju Island, Korea.

George Karypis. 2003. CLUTO: A clustering toolkit. Technical Report 02-017, University of Minnesota Department of Computer Science.

Sophia Katrenko and Pieter Adriaans. 2008. Qualia structures and their impact on the concrete noun categorization task. In *Proceedings of the ESSLLI Workshop on Distributional Lexical Semantics*, pages 17–24, Hamburg, Germany.

Thomas Landauer and Susan Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240.

Daniel Lee and Sebastian Seung. 2000. Algorithms for Non-negative Matrix Factorization. In *Proceedings of NIPS*, pages 556–562.

Chih-Jen Lin. 2007. Projected gradient methods for Nonnegative Matrix Factorization. *Neural Computation*, 19(10):2756–2779.

Ken McRae, Michael Spivey-Knowlton, and Michael Tanenhaus. 1998. Modeling the influence of thematic fit (and other constraints) in on-line sentence comprehension. *Journal of Memory and Language*, 38:283–312.

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

Tomas Mikolov, Quoc Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for Machine Translation. `http://arxiv.org/abs/1309.4168`.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeff Dean. 2013c. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119, Lake Tahoe, Nevada.

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

George Miller and Walter Charles. 1991. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28.

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

Ulrike Padó. 2007. *The Integration of Syntax and Semantic Plausibility in a Wide-Coverage Model of Sentence Processing*. Dissertation, Saarland University, Saarbrücken.

Klaus Rothenhäusler and Hinrich Schütze. 2009. Unsupervised classification with dependency based word spaces. In *Proceedings of GEMS*, pages 17–24, Athens, Greece.

Herbert Rubenstein and John Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Magnus Sahlgren. 2006. *The Word-Space Model*. Ph.D dissertation, Stockholm University.

Richard Socher, Brody Huval, Christopher Manning, and Andrew Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP*, pages 1201–1211, Jeju Island, Korea.

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

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of ACL*, pages 384–394, Uppsala, Sweden.

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

Peter Turney. 2012. Domain and function: A dual-space model of semantic relations and compositions. *Journal of Artificial Intelligence Research*, 44:533–585.

Metaphor Detection with Cross-Lingual Model Transfer

Yulia Tsvetkov Leonid Boytsov Anatole Gershman Eric Nyberg Chris Dyer
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213 USA
{ytsvetko, srchvrs, anatoleg, ehn, cdyer}@cs.cmu.edu

Abstract

We show that it is possible to reliably discriminate whether a syntactic construction is meant literally or metaphorically using lexical semantic features of the words that participate in the construction. Our model is constructed using English resources, and we obtain state-of-the-art performance relative to previous work in this language. Using a model transfer approach by pivoting through a bilingual dictionary, we show our model can identify metaphoric expressions in other languages. We provide results on three new test sets in Spanish, Farsi, and Russian. The results support the hypothesis that metaphors are conceptual, rather than lexical, in nature.

1 Introduction

Lakoff and Johnson (1980) characterize metaphor as reasoning about one thing in terms of another, i.e., a metaphor is a type of *conceptual mapping*, where words or phrases are applied to objects and actions in ways that do not permit a literal interpretation. They argue that metaphors play a fundamental communicative role in verbal and written interactions, claiming that much of our everyday language is delivered in metaphorical terms. There is empirical evidence supporting the claim: recent corpus studies have estimated that the proportion of words used metaphorically ranges from 5% to 20% (Steen et al., 2010), and Thibodeau and Boroditsky (2011) provide evidence that a choice of metaphors affects decision making.

Given the prevalence and importance of metaphoric language, effective automatic detection of metaphors would have a number of benefits, both practical and scientific. Language processing applications that need to understand language or preserve meaning (information extraction, machine translation, dialog systems, sentiment analysis, and text analytics, etc.) would have access to a potentially useful high-level bit of information about whether something is to be understood literally or not. Second, scientific hypotheses about metaphoric language could be tested more easily at a larger scale with automation.

However, metaphor detection is a hard problem. On one hand, there is a subjective component: humans may disagree whether a particular expression is used metaphorically or not, as there is no clear-cut semantic distinction between figurative and metaphorical language (Shutova, 2010). On the other, metaphors can be domain- and context-dependent.[1]

Previous work has focused on metaphor identification in English, using both extensive manually-created linguistic resources (Mason, 2004; Gedigian et al., 2006; Krishnakumaran and Zhu, 2007; Turney et al., 2011; Broadwell et al., 2013) and corpus-based approaches (Birke and Sarkar, 2007; Shutova et al., 2013; Neuman et al., 2013; Shutova and Sun, 2013; Hovy et al., 2013). We build on this foundation and also extend metaphor detection into other languages in which few resources may exist. Our work makes the following contributions: (1) we develop a new state-of-the-art English metaphor detection system that uses *conceptual* semantic features, such as a degree of abstractness and semantic supersenses;[2] (2) we create new metaphor-annotated corpora for Russian and English;[3] (3) using a paradigm of model transfer (McDonald et al., 2011; Täckström et al., 2013; Kozhenikov and Titov, 2013), we provide support for the hypothesis that metaphors are concep-

[1] For example, *drowning students* could be used metaphorically to describe the situation where students are overwhelmed with work, but in the sentence *a lifeguard saved drowning students*, this phrase is used literally.

[2] https://github.com/ytsvetko/metaphor

[3] http://www.cs.cmu.edu/~ytsvetko/metaphor/datasets.zip

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 248–258,
Baltimore, Maryland, USA, June 23-25 2014.

tual (rather than lexical) in nature by showing that our English-trained model can detect metaphors in Spanish, Farsi, and Russian.

2 Methodology

Our task in this work is to define features that distinguish between metaphoric and literal uses of two syntactic constructions: subject-verb-object (SVO) and adjective-noun (AN) tuples.[4] We give examples of a prototypical metaphoric usage of each type:

- **SVO metaphors.** A sentence containing a metaphoric SVO relation is *my car drinks gasoline*. According to Wilks (1978), this metaphor represents a violation of selectional preferences for the verb *drink*, which is normally associated with animate subjects (the car is inanimate and, hence, cannot drink in the literal sense of the verb).

- **AN metaphors.** The phrase *broken promise* is an AN metaphor, where attributes from a concrete domain (associated with the concrete word *broken*) are transferred to a more abstract domain, which is represented by the relatively abstract word *promise*. That is, we map an abstract concept *promise* to a concrete domain of physical things, where things can be literally broken to pieces.

Motivated by Lakoff's (1980) argument that metaphors are systematic conceptual mappings, we will use coarse-grained *conceptual*, rather than fine-grained *lexical* features, in our classifier. Conceptual features pertain to concepts and ideas as opposed to individual words or phrases expressed in a particular language. In this sense, as long as two words in two different languages refer to the same concepts, their conceptual features should be the same. Furthermore, we hypothesize that our coarse semantic features give us a language-invariant representation suitable for metaphor detection. To test this hypothesis, we use a cross-lingual model transfer approach: we use bilingual dictionaries to project words from other syntactic constructions found in other languages into English and then apply the English model on the derived conceptual representations.

[4]Our decision to focus on SVO and AN metaphors is justified by corpus studies that estimate that verb- and adjective-based metaphors account for a substantial proportion of all metaphoric expressions, approximately 60% and 24%, respectively (Shutova and Teufel, 2010; Gandy et al., 2013).

Each SVO (or AN) instance will be represented by a triple (duple) from which a feature vector will be extracted.[5] The vector will consist of the concatenation of the conceptual features (which we discuss below) for all participating words, and conjunction features for word pairs.[6] For example, to generate the feature vector for the SVO triple $(car, drink, gasoline)$, we compute all the features for the individual words *car*, *drink*, *gasoline* and combine them with the conjunction features for the pairs *car drink* and *drink gasoline*.

We define three main feature categories (1) abstractness and imageability, (2) supersenses, (3) unsupervised vector-space word representations; each category corresponds to a group of features with a common theme and representation.

- **Abstractness and imageability.** Abstractness and imageability were shown to be useful in detection of metaphors (it is easier to invoke mental pictures of concrete and imageable words) (Turney et al., 2011; Broadwell et al., 2013). We expect that abstractness, used in conjunction features (e.g., a feature denoting that the subject is abstract and the verb is concrete), is especially useful: semantically, an abstract agent performing a concrete action is a strong signal of metaphorical usage.

  Although often correlated with abstractness, imageability is not a redundant property. While most abstract things are hard to visualize, some call up images, e.g., *vengeance* calls up an emotional image, *torture* calls up emotions and even visual images. There are concrete things that are hard to visualize too, for example, *abbey* is harder to visualize than *banana* (B. MacWhinney, personal communication).

- **Supersenses.** Supersenses[7] are coarse semantic categories originating in WordNet. For nouns and verbs there are 45 classes: 26 for nouns and 15 for verbs, for example,

[5]Looking at components of the syntactic constructions independent of their context has its limitations, as discussed above with the *drowning students* example; however, it simplifies the representation challenges considerably.

[6]If word one is represented by features $\mathbf{u} \in \mathbb{R}^n$ and word two by features $\mathbf{v} \in \mathbb{R}^m$ then the conjunction feature vector is the vectorization of the outer product $\mathbf{u}\mathbf{v}^\top$.

[7]Supersenses are called "lexicographer classes" in WordNet documentation (Fellbaum, 1998), `http://wordnet.princeton.edu/man/lexnames.5WN.html`

noun.body, *noun.animal*, *verb.consumption*, or *verb.motion* (Ciaramita and Altun, 2006). English adjectives do not, as yet, have a similar high-level semantic partitioning in WordNet, thus we use a 13-class taxonomy of adjective supersenses constructed by Tsvetkov et al. (2014) (discussed in §3.2).

Supersenses are particularly attractive features for metaphor detection: coarse sense taxonomies can be viewed as semantic concepts, and since concept mapping is a process in which metaphors are born, we expect different supersense co-occurrences in metaphoric and literal combinations. In "drinks gasoline", for example, mapping to supersenses would yield a pair <*verb.consumption, noun.substance*>, contrasted with <*verb.consumption, noun.food*> for "drinks juice". In addition, this coarse semantic categorization is preserved in translation (Schneider et al., 2013), which makes supersense features suitable for cross-lingual approaches such as ours.

- **Vector space word representations.** Vector space word representations learned using unsupervised algorithms are often effective features in supervised learning methods (Turian et al., 2010). In particular, many such representations are designed to capture lexical semantic properties and are quite effective features in semantic processing, including named entity recognition (Turian et al., 2009), word sense disambiguation (Huang et al., 2012), and lexical entailment (Baroni et al., 2012). In a recent study, Mikolov et al. (2013) reveal an interesting cross-lingual property of distributed word representations: there is a strong similarity between the vector spaces across languages that can be easily captured by linear mapping. Thus, vector space models can also be seen as vectors of (latent) semantic concepts, that preserve their "meaning" across languages.

3 Model and Feature Extraction

In this section we describe a classification model, and provide details on mono- and cross-lingual implementation of features.

3.1 Classification using Random Forests

To make classification decisions, we use a random forest classifier (Breiman, 2001), an ensemble of decision tree classifiers learned from many independent subsamples of the training data. Given an input, each tree classifier assigns a probability to each label; those probabilities are averaged to compute the probability distribution across the ensemble. Random forest ensembles are particularly suitable for our resource-scarce scenario: rather than overfitting, they produce a limiting value of the generalization error as the number of trees increases,[8] and no hyperparameter tuning is required. In addition, decision-tree classifiers learn non-linear responses to inputs and often outperform logistic regression (Perlich et al., 2003).[9] Our random forest classifier models the probability that the input syntactic relation is metaphorical. If this probability is above a threshold, the relation is classified as metaphoric, otherwise it is literal. We used the `scikit-learn` toolkit to train our classifiers (Pedregosa et al., 2011).

3.2 Feature extraction

Abstractness and imageability. The MRC psycholinguistic database is a large dictionary listing linguistic and psycholinguistic attributes obtained experimentally (Wilson, 1988).[10] It includes, among other data, 4,295 words rated by the degrees of abstractness and 1,156 words rated by the imageability. Similarly to Tsvetkov et al. (2013), we use a logistic regression classifier to propagate abstractness and imageability scores from MRC ratings to all words for which we have vector space representations. More specifically, we calculate the degree of abstractness and imageability of all English items that have a vector space representation, using vector elements as features. We train two separate classifiers for abstractness and imageability on a seed set of words from the MRC database. Degrees of abstractness and imageability are posterior probabilities of classifier predictions. We binarize these posteriors into abstract-concrete (or imageable-unimageable) boolean indicators using pre-defined thresholds.[11] Perfor-

[8] See Theorem 1.2 in (Breiman, 2001) for details.

[9] In our experiments, random forests model slightly outperformed logistic regression and SVM classifiers.

[10] `http://ota.oucs.ox.ac.uk/headers/1054.xml`

[11] Thresholds are equal to 0.8 for abstractness and to 0.9 for imageability. They were chosen empirically based on ac-

mance of these classifiers, tested on a sampled held-out data, is 0.94 and 0.85 for the abstractness and imageability classifiers, respectively.

Supersenses. In the case of SVO relations, we incorporate supersense features for nouns and verbs; noun and adjective supersenses are used in the case of AN relations.

Supersenses of nouns and verbs. A lexical item can belong to several synsets, which are associated with different supersenses. Degrees of membership in different supersenses are represented by feature vectors, where each element corresponds to one supersense. For example, the word *head* (when used as a noun) participates in 33 synsets, three of which are related to the supersense *noun.body*. The value of the feature corresponding to this supersense is $3/33 \approx 0.09$.

Supersenses of adjectives. WordNet lacks coarse-grained semantic categories for adjectives. To divide adjectives into groups, Tsvetkov et al. (2014) use 13 top-level classes from the adapted taxonomy of Hundsnurscher and Splett (1982), which is incorporated in GermaNet (Hamp and Feldweg, 1997). For example, the top-level classes in GermaNet include: *adj.feeling* (e.g., willing, pleasant, cheerful); *adj.substance* (e.g., dry, ripe, creamy); *adj.spatial* (e.g., adjacent, gigantic).[12] For each adjective type in WordNet, they produce a vector with a classifier posterior probabilities corresponding to degrees of membership of this word in one of the 13 semantic classes,[13] similar to the feature vectors we build for nouns and verbs. For example, for a word *calm* the top-2 categories (with the first and second highest degrees of membership) are *adj.behavior* and *adj.feeling*.

Vector space word representations. We employ 64-dimensional vector-space word representations constructed by Faruqui and Dyer (2014).[14] Vector construction algorithm is a variation on traditional latent semantic analysis (Deerwester et al., 1990) that uses multilingual information to produce representations in which synonymous words have similar vectors. The vectors were trained on the news commentary corpus released by WMT-2011,[15] comprising 180,834 types.

3.3 Cross-lingual feature projection

For languages other than English, feature vectors are projected to English features using translation dictionaries. We used the Babylon dictionary,[16] which is a proprietary resource, but any bilingual dictionary can in principle be used. For a non-English word in a source language, we first obtain all translations into English. Then, we average all feature vectors related to these translations. Consider an example related to projection of WordNet supersenses. A Russian word голова is translated as *head* and *brain*. Hence, we select all the synsets of the nouns *head* and *brain*. There are 38 such synsets (33 for *head* and 5 for *brain*). Four of these synsets are associated with the supersense *noun.body*. Therefore, the value of the feature *noun.body* is $4/38 \approx 0.11$.

4 Datasets

In this section we describe a training and testing dataset as well a data collection procedure.

4.1 English training sets

To train an SVO metaphor classifier, we employ the TroFi (Trope Finder) dataset.[17] TroFi includes 3,737 manually annotated English sentences from the *Wall Street Journal* (Birke and Sarkar, 2007). Each sentence contains either literal or metaphorical use for one of 50 English verbs. First, we use a dependency parser (Martins et al., 2010) to extract subject-verb-object (SVO) relations. Then, we filter extracted relations to eliminate parsing-related errors, and relations with verbs which are not in the TroFi verb list. After filtering, there are 953 metaphorical and 656 literal SVO relations which we use as a training set.

In the case of AN relations, we construct and make publicly available a training set containing 884 metaphorical AN pairs and 884 pairs with literal meaning. It was collected by two annotators using public resources (collections of metaphors on the web). At least one additional person carefully examined and culled the collected metaphors, by removing duplicates, weak metaphors, and metaphorical phrases (such as

curacy during cross-validation.

[12] For the full taxonomy see http://www.sfs.uni-tuebingen.de/lsd/adjectives.shtml

[13] http://www.cs.cmu.edu/~ytsvetko/adj-supersenses.tar.gz

[14] http://www.cs.cmu.edu/~mfaruqui/soft.html

[15] http://www.statmt.org/wmt11/

[16] http://www.babylon.com

[17] http://www.cs.sfu.ca/~anoop/students/jbirke/

drowning students) whose interpretation depends on the context.

4.2 Multilingual test sets

We collect and annotate metaphoric and literal test sentences in four languages. Thus, we compile eight test datasets, four for SVO relations, and four for AN relations. Each dataset has an equal number of metaphors and non-metaphors, i.e., the datasets are balanced. English (EN) and Russian (RU) datasets have been compiled by our team and are publicly available. Spanish (ES) and Farsi (FA) datasets are published elsewhere (Levin et al., 2014). Table 1 lists test set sizes.

	SVO	AN
EN	222	200
RU	240	200
ES	220	120
FA	44	320

Table 1: Sizes of the eight test sets. Each dataset is balanced, i.e., it has an equal number of metaphors and non-metaphors. For example, English SVO dataset has 222 relations: 111 metaphoric and 111 literal.

We used the following procedure to compile the EN and RU test sets. A moderator started with seed lists of 1000 most common verbs and adjectives.[18]

Then she used the SketchEngine, which provides searching capability for the TenTen Web corpus,[19] to extract sentences with words that frequently co-occurred with words from the seed lists. From these sentences, she removed sentences that contained more than one metaphor, and sentences with non-SVO and non-AN metaphors. Remaining sentences were annotated by several native speakers (five for English and six for Russian), who judged AN and SVO phrases in context. The annotation instructions were general: *"Please, mark in bold all words that, in your opinion, are used non-literally in the following sentences. In many sentences, all the words may be used literally."* The Fleiss' Kappas for 5 English and 6 Russian annotators are: EN-AN = .76, RU-AN = .85, EN-SVO = .75, RU-SVO = .78. For the final selection, we filtered out low-agreement ($<.8$) sentences.

The test candidate sentences were selected by a person who did not participate in the selection of the training samples. No English annotators of the test set, and only one Russian annotator out of 6 participated in the selection of the training samples. Thus, we trust that annotator judgments were not biased towards the cases that the system is trained to process.

5 Experiments

5.1 English experiments

Our task, as defined in Section 2, is to classify SVO and AN relations as either metaphoric or literal. We first conduct a 10-fold cross-validation experiment on the training set defined in Section 4.1. We represent each candidate relation using the features described in Section 3.2, and evaluate performance of the three feature categories and their combinations. This is done by computing an accuracy in the 10-fold cross validation. Experimental results are given in Table 2, where we also provide the number of features in each feature set.

	SVO		AN	
	# FEAT	ACC	# FEAT	ACC
AbsImg	20	0.73*	16	0.76*
Supersense	67	0.77*	116	0.79*
AbsImg+Sup.	87	0.78*	132	0.80*
VSM	192	0.81	228	0.84*
All	279	**0.82**	360	**0.86**

Table 2: 10-fold cross validation results for three feature categories and their combination, for classifiers trained on English SVO and AN training sets. # FEAT column shows a number of features. ACC column reports an accuracy score in the 10-fold cross validation. Statistically significant differences ($p < 0.01$) from the all-feature combination are marked with a star.

These results show superior performance over previous state-of-the-art results, confirming our hypothesis that conceptual features are effective in metaphor classification. For the SVO task, the cross-validation accuracy is about 10% better than that of Tsvetkov et al. (2013). For the AN task, the cross validation accuracy is better by 8% than the result of Turney et al. (2011) (two baseline

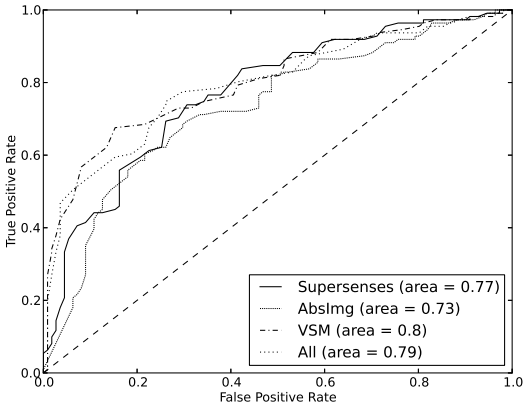
[18] Selection of 1000 most common verbs and adjectives achieves much broader lexical and domain coverage than what can be realistically obtained from continuous text. Our test sentence domains are, therefore, diverse: economic, political, sports, etc.

[19] http://trac.sketchengine.co.uk/wiki/Corpora/enTenTen

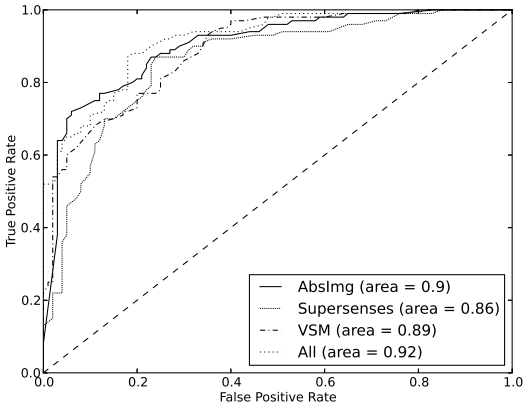
methods are described in Section 5.2). We can see that all types of features have good performance on their own (VSM is the strongest feature type). Noun supersense features alone allows us to achieve an accuracy of 75%, i.e., adjective supersense features contribute 4% to adjective-noun supersense feature combination. Experiments with the pairs of features yield better results than individual features, implying that the feature categories are not redundant. Yet, combining all features leads to even higher accuracy during cross-validation. In the case of the AN task, a difference between the All feature combination and any other combination of features listed in Table 2 is statistically significant ($p < 0.01$ for both the sign and the permutation test).

Although the first experiment shows very high scores, the 10-fold cross-validation cannot fully reflect the generality of the model, because all folds are parts of the same corpus. They are collected by the same human judges and belong to the same domain. Therefore, experiments on out-of-domain data are crucial. We carry out such experiments using held-out SVO and AN EN test sets, described in Section 4.2 and Table 1. In this experiment, we measure the f-score. We classify SVO and AN relations using a classifier trained on the All feature combination and balanced thresholds. The values of the f-score are 0.76, both for SVO and AN tasks. This out-of-domain experiment suggests that our classifier is portable across domains and genres.

However, (1) different application may have different requirements for recall/precision, and (2) classification results may be skewed towards having high precision and low recall (or vice versa). It is possible to trade precision for recall by choosing a different threshold. Thus, in addition to giving a single f-score value for balanced thresholds, we present a Receiver Operator Characteristic (ROC) curve, where we plot a fraction of true positives against the fraction of false positives for 100 threshold values in the range from zero to one. The area under the ROC curve (AUC) can be interpreted as the probability that a classifier will assign a higher score to a randomly chosen positive example than to a randomly chosen negative example.[20] For a randomly guessing classifier, the ROC curve is a dashed diagonal line. A bad classifier has an ROC curve that goes close to the dashed diagonal or even below it.



(a) **SVO**



(b) **AN**

Figure 1: ROC curves for classifiers trained using different feature sets (English SVO and AN test sets).

According to ROC plots in Figure 1, all three feature sets are effective, both for SVO and for AN tasks. Abstractness and Imageability features work better for adjectives and nouns, which is in line with previous findings (Turney et al., 2011; Broadwell et al., 2013). It can be also seen that VSM features are very effective. This is in line with results of Hovy et al. (2013), who found that it is hard to improve over the classifier that uses only VSM features.

5.2 Comparison to baselines

In this section, we compare our method to state-of-the-art methods of Tsvetkov et al. (2013) and of Turney et al. (2011), who focused on classifying SVO and AN relations, respectively.

In the case of SVO relations, we use software

[20] Assuming that positive examples are labeled by ones, and negative examples are labeled by zeros.

and datasets from Tsvetkov et al. (2013). These datasets, denoted as an SVO-baseline, consist of 98 English and 149 Russian sentences. We train SVO metaphor detection tools on SVO relations extracted from TroFi sentences and evaluate them on the SVO-baseline dataset. We also use the same thresholds for classifier posterior probabilities as Tsvetkov et al. (2013). Our approach is different from that of Tsvetkov et al. (2013) in that it uses additional features (vector space word representations) and a different classification method (we use random forests while Tsvetkov et al. (2013) use logistic regression). According to Table 3, we obtain higher performance scores for both Russian and English.

	EN	**RU**
SVO-baseline	0.78	0.76
This work	0.86	0.85

Table 3: Comparing f-scores of our SVO metaphor detection method to the baselines.

In the case of AN relations, we use the dataset (denoted as an AN-baseline) created by Turney et al. (2011) (see Section 4.1 in the referred paper for details). Turney et al. (2011) manually annotated 100 pairs where an adjective was one of the following: *dark*, *deep*, *hard*, *sweet*, and *worm*. The pairs were presented to five human judges who rated each pair on a scale from 1 (very literal/denotative) to 4 (very non-literal/connotative). Turney et al. (2011) train logistic-regression employing only abstractness ratings as features. Performance of the method was evaluated using the 10-fold cross-validation separately for each judge.

We replicate the above described evaluation procedure of Turney et al. (2011) using their model and features. In our classifier, we use the All feature combination and the balanced threshold as described in Section 5.1.

According to results in Table 4, almost all of the judge-specific f-scores are slightly higher for our system, as well as the overall average f-score.

In both baseline comparisons, we obtain performance at least as good as in previously published studies.

5.3 Cross-lingual experiments

In the next experiment we corroborate the main hypothesis of this paper: a model trained on English data can be successfully applied to other languages. Namely, we use a trained English model discussed in Section 5.1 to classify literal and metaphoric SVO and AN relations in English, Spanish, Farsi and Russian test sets, listed in Section 4.2. This time we used all available features.

	AN-baseline	**This work**
Judge 1	0.73	0.75
Judge 2	0.81	0.84
Judge 3	0.84	0.88
Judge 4	0.79	0.81
Judge 5	0.78	0.77
average	0.79	0.81

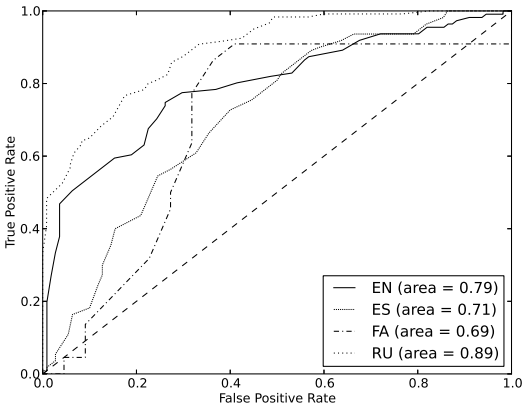
Table 4: Comparing AN metaphor detection method to the baselines: accuracy of the 10-fold cross validation on annotations of five human judges.

Experimental results for all four languages, are given in Figure 2. The ROC curves for SVO and AN tasks are plotted in Figure 2a and Figure 2b, respectively. Each curve corresponds to a test set described in Table 1. In addition, we perform an oracle experiment, to obtain actual f-score values for best thresholds. Detailed results are shown in Table 5.

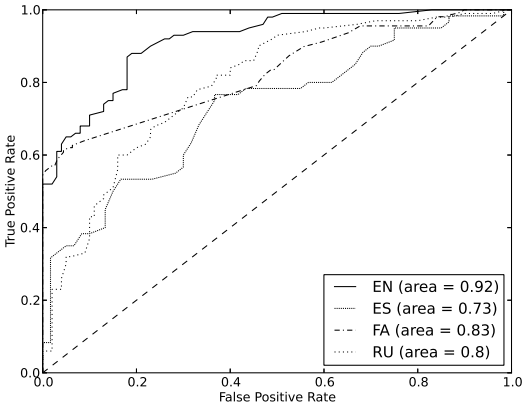
Consistent results with high f-scores are obtained across all four languages. Note that higher scores are obtained for the Russian test set. We hypothesize that this happens due to a higher-quality translation dictionary (which allows a more accurate model transfer). Relatively lower (yet reasonable) results for Farsi can be explained by a smaller size of the bilingual dictionary (thus, fewer feature projections can be obtained). Also note that, in our experience, most of Farsi metaphors are adjective-noun constructions. This is why the AN FA dataset in Table 1 is significantly larger than SVO FA. In that, for the AN Farsi task we observe high performance scores.

Figure 2 and Table 5 confirm, that *we obtain similar, robust results on four very different languages, using the same English classifiers*. We view this result as a strong evidence of language-independent nature of our metaphor detection method. In particular, this shows that proposed conceptual features can be used to detect selectional preferences violation across languages.

To summarize the experimental section, our metaphor detection approach obtains state-of-the-



(a) **SVO**



(b) **AN**

Figure 2: Cross-lingual experiment: ROC curves for classifiers trained on the English data using a combination of all features, and applied to SVO and AN metaphoric and literal relations in four test languages: English, Russian, Spanish, and Farsi.

art performance in English, is effective when applied to out-of-domain English data, and works cross-lingually.

5.4 Examples

Manual data analysis on adjective-noun pairs supports an abstractness-concreteness hypothesis formulated by several independent research studies. For example, in English we classify as metaphoric *dirty word* and *cloudy future*. Word pairs *dirty diaper* and *cloudy weather* have same adjectives. Yet they are classified as literal. Indeed, *diaper* is a more concrete term than *word* and *weather* is more concrete than *future*. Same pattern is observed in non-English datasets. In Russian, больное общество "sick society" and пустой звук "empty sound" are classified as metaphoric, while

	SVO	AN
EN	0.79	0.85
RU	0.84	0.77
ES	0.76	0.72
FA	0.75	0.74

Table 5: Cross-lingual experiment: f-scores for classifiers trained on the English data using a combination of all features, and applied, with optimal thresholds, to SVO and AN metaphoric and literal relations in four test languages: English, Russian, Spanish, and Farsi.

больная бабушка "sick grandmother" and пустая чашка "empty cup" are classified as literal. Spanish example of an adjective-noun metaphor is a well-known *músculo económico* "economic muscle". We also observe that non-metaphoric adjective noun pairs tend to have more imageable adjectives, such as literal *derecho humano* "human right". In Spanish, *human* is more imageable than *economic*.

Verb-based examples that are correctly classified by our model are: *blunder escaped notice* (metaphoric) and *prisoner escaped jail* (literal). We hypothesize that supersense features are instrumental in the correct classification of these examples: <*noun.person,verb.motion*> is usually used literally, while <*noun.act,verb.motion*> is used metaphorically.

6 Related Work

For a historic overview and a survey of common approaches to metaphor detection, we refer the reader to recent reviews by Shutova et al. (Shutova, 2010; Shutova et al., 2013). Here we focus only on recent approaches.

Shutova et al. (2010) proposed a bottom-up method: one starts from a set of seed metaphors and seeks phrases where verbs and/or nouns belong to the same cluster as verbs or nouns in seed examples.

Turney et al. (2011) show how abstractness scores could be used to detect metaphorical AN phrases. Neuman et al. (2013) describe a Concrete Category Overlap algorithm, where co-occurrence statistics and Turney's abstractness scores are used to determine WordNet supersenses that correspond to literal usage of a given adjective or verb. For example, given an adjective, we can learn that it modifies concrete nouns that usually have the

supersense *noun.body*. If this adjective modifies a noun with the supersense *noun.feeling*, we conclude that a metaphor is found.

Broadwell et al. (2013) argue that metaphors are highly imageable words that do not belong to a discussion topic. To implement this idea, they extend MRC imageability scores to all dictionary words using links among WordNet supersenses (mostly hypernym and hyponym relations). Strzalkowski et al. (2013) carry out experiments in a specific (government-related) domain for four languages: English, Spanish, Farsi, and Russian. Strzalkowski et al. (2013) explain the algorithm only for English and say that is the same for Spanish, Farsi, and Russian. Because they heavily rely on WordNet and availability of imageability scores, their approach may not be applicable to low-resource languages.

Hovy et al. (2013) applied tree kernels to metaphor detection. Their method also employs WordNet supersenses, but it is not clear from the description whether WordNet is essential or can be replaced with some other lexical resource. We cannot compare directly our model with this work because our classifier is restricted to detection of only SVO and AN metaphors.

Tsvetkov et al. (2013) propose a cross-lingual detection method that uses only English lexical resources and a dependency parser. Their study focuses only on the verb-based metaphors. Tsvetkov et al. (2013) employ only English and Russian data. Current work builds on this study, and incorporates new syntactic relations as metaphor candidates, adds several new feature sets and different, more reliable datasets for evaluating results. We demonstrate results on two new languages, Spanish and Farsi, to emphasize the generality of the method.

A words sense disambiguation (WSD) is a related problem, where one identifies meanings of polysemous words. The difference is that in the WSD task, we need to select an already existing sense, while for the metaphor detection, the goal is to identify cases of sense borrowing. Studies showed that cross-lingual evidence allows one to achieve a state-of-the-art performance in the WSD task, yet, most cross-lingual WSD methods employ parallel corpora (Navigli, 2009).

7 Conclusion

The key contribution of our work is that we show how to identify metaphors across languages by building a model in English and applying it—without adaptation—to other languages: Spanish, Farsi, and Russian. This model uses language-independent (rather than lexical or language specific) conceptual features. Not only do we establish benchmarks for Spanish, Farsi, and Russian, but we also achieve state-of-the-art performance in English. In addition, we present a comparison of relative contributions of several types of features. We concentrate on metaphors in the context of two kinds of syntactic relations: subject-verb-object (SVO) relations and adjective-noun (AN) relations, which account for a majority of all metaphorical phrases.

Future work will expand the scope of metaphor identification by including nominal metaphoric relations as well as explore techniques for incorporating contextual features, which can play a key role in identifying certain kinds of metaphors. Second, cross-lingual model transfer can be improved with more careful cross-lingual feature projection.

Acknowledgments

We are extremely grateful to Shuly Wintner for a thorough review that helped us improve this draft; we also thank people who helped in creating the datasets and/or provided valuable feedback on this work: Ed Hovy, Vlad Niculae, Davida Fromm, Brian MacWhinney, Carlos Ramírez, and other members of the CMU METAL team. This work was supported by the U.S. Army Research Laboratory and the U.S. Army Research Office under contract/grant number W911NF-10-1-0533.

References

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

Julia Birke and Anoop Sarkar. 2007. Active learning for the identification of nonliteral language. In *Proc. of the Workshop on Computational Approaches to Figurative Language*, FigLanguages '07, pages 21–28.

Leo Breiman. 2001. Random forests. *Machine Learning*, 45(1):5–32.

George Aaron Broadwell, Umit Boz, Ignacio Cases, Tomek Strzalkowski, Laurie Feldman, Sarah Taylor, Samira Shaikh, Ting Liu, Kit Cho, and Nick Webb. 2013. Using imageability and topic chaining to locate metaphors in linguistic corpora. In *Social Computing, Behavioral-Cultural Modeling and Prediction*, pages 102–110. Springer.

Massimiliano Ciaramita and Yasemin Altun. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proc. of EMNLP*, pages 594–602.

Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proc. of EACL*. Association for Computational Linguistics.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. Language, Speech and Communication. MIT Press.

Lisa Gandy, Nadji Allan, Mark Atallah, Ophir Frieder, Newton Howard, Sergey Kanareykin, Moshe Koppel, Mark Last, Yair Neuman, and Shlomo Argamon. 2013. Automatic identification of conceptual metaphors with limited knowledge. In *Proc. of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, pages 328–334.

Matt Gedigian, John Bryant, Srini Narayanan, and Branimir Ciric. 2006. Catching metaphors. In *Proceedings of the 3rd Workshop on Scalable Natural Language Understanding*, pages 41–48.

Birgit Hamp and Helmut Feldweg. 1997. Germanet-a lexical-semantic net for German. In *Proc. of ACL workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, pages 9–15.

Dirk Hovy, Shashank Srivastava, Sujay Kumar Jauhar, Mrinmaya Sachan, Kartik Goyal, Huiying Li, Whitney Sanders, and Eduard Hovy. 2013. Identifying metaphorical word use with tree kernels. In *Proc. of the First Workshop on Metaphor in NLP*, page 52.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proc. of ACL*, pages 873–882.

Franz Hundsnurscher and Jochen Splett. 1982. *Semantik der Adjektive des Deutschen*. Number 3137. Westdeutscher Verlag.

Mikhail Kozhenikov and Ivan Titov. 2013. Cross-lingual transfer of semantic role labeling models. In *Proc. of ACL*, pages 1190–1200.

Saisuresh Krishnakumaran and Xiaojin Zhu. 2007. Hunting elusive metaphors using lexical resources. In *Proc. of the Workshop on Computational approaches to Figurative Language*, pages 13–20.

George Lakoff and Mark Johnson. 1980. Conceptual metaphor in everyday language. *The Journal of Philosophy*, pages 453–486.

Lori Levin, Teruko Mitamura, Davida Fromm, Brian MacWhinney, Jaime Carbonell, Weston Feely, Robert Frederking, Anatole Gershman, and Carlos Ramirez. 2014. Resources for the detection of conventionalized metaphors in four languages. In *Proc. of LREC*.

André F. T. Martins, Noah A. Smith, Eric P. Xing, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. 2010. Turbo parsers: dependency parsing by approximate variational inference. In *Proc. of ENMLP*, pages 34–44.

Zachary J Mason. 2004. CorMet: a computational, corpus-based conventional metaphor extraction system. *Computational Linguistics*, 30(1):23–44.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proc. of EMNLP*.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for Machine Translation. *CoRR*, abs/1309.4168.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2):10:1–10:69, February.

Yair Neuman, Dan Assaf, Yohai Cohen, Mark Last, Shlomo Argamon, Newton Howard, and Ophir Frieder. 2013. Metaphor identification in large texts corpora. *PloS one*, 8(4):e62343.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Claudia Perlich, Foster Provost, and Jeffrey S. Simonoff. 2003. Tree induction vs. logistic regression: a learning-curve analysis. *Journal of Machine Learning Research*, 4:211–255.

Nathan Schneider, Behrang Mohit, Chris Dyer, Kemal Oflazer, and Noah A Smith. 2013. Supersense tagging for Arabic: the MT-in-the-middle attack. In *Proc. of NAACL-HLT*, pages 661–667.

Ekaterina Shutova and Lin Sun. 2013. Unsupervised metaphor identification using hierarchical graph factorization clustering. In *Proc. of NAACL-HLT*, pages 978–988.

Ekaterina Shutova and Simone Teufel. 2010. Metaphor corpus annotated for source-target domain mappings. In *Proc. of LREC*, pages 3255–3261.

Ekaterina Shutova, Lin Sun, and Anna Korhonen. 2010. Metaphor identification using verb and noun clustering. In *Proc. of COLING*, pages 1002–1010.

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

Ekaterina Shutova. 2010. Models of metaphor in NLP. In *Proc. of ACL*, pages 688–697.

Gerard J Steen, Aletta G Dorst, J Berenike Herrmann, Anna A Kaal, and Tina Krennmayr. 2010. Metaphor in usage. *Cognitive Linguistics*, 21(4):765–796.

Tomek Strzalkowski, George Aaron Broadwell, Sarah Taylor, Laurie Feldman, Boris Yamrom, Samira Shaikh, Ting Liu, Kit Cho, Umit Boz, Ignacio Cases, et al. 2013. Robust extraction of metaphors from novel data. In *Proc. of the First Workshop on Metaphor in NLP*, page 67.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. *TACL*, 1:1–12.

Paul H Thibodeau and Lera Boroditsky. 2011. Metaphors we think with: The role of metaphor in reasoning. *PLoS One*, 6(2):e16782.

Yulia Tsvetkov, Elena Mukomel, and Anatole Gershman. 2013. Cross-lingual metaphor detection using common semantic features. In *The 1st Workshop on Metaphor in NLP 2013*, page 45.

Yulia Tsvetkov, Nathan Schneider, Dirk Hovy, Archna Bhatia, Manaal Faruqui, and Chris Dyer. 2014. Augmenting English adjective senses with supersenses. In *Proc. of LREC*.

Joseph Turian, Lev Ratinov, Yoshua Bengio, and Dan Roth. 2009. A preliminary evaluation of word representations for named-entity recognition. In *NIPS Workshop on Grammar Induction, Representation of Language and Language Learning*, pages 1–8.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proc. of ACL*, pages 384–394.

Peter D. Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. Literal and metaphorical sense identification through concrete and abstract context. In *Proc. of EMNL*, pages 680–690.

Yorick Wilks. 1978. Making preferences more active. *Artificial Intelligence*, 11(3):197–223.

Michael Wilson. 1988. MRC Psycholinguistic Database: Machine-usable dictionary, version 2.00. *Behavior Research Methods, Instruments, & Computers*, 20(1):6–10.

Learning Word Sense Distributions, Detecting Unattested Senses and Identifying Novel Senses Using Topic Models

Jey Han Lau,[♠] **Paul Cook,**[♡] **Diana McCarthy,**[♢] **Spandana Gella,**[♡] and **Timothy Baldwin**[♡]
♠ Dept of Philosophy, King's College London
♡ Dept of Computing and Information Systems, The University of Melbourne
♢ University of Cambridge
jeyhan.lau@gmail.com, paulcook@unimelb.edu.au,
diana@dianamccarthy.co.uk, spandanagella@gmail.com, tb@ldwin.net

Abstract

Unsupervised word sense disambiguation (WSD) methods are an attractive approach to all-words WSD due to their non-reliance on expensive annotated data. Unsupervised estimates of sense frequency have been shown to be very useful for WSD due to the skewed nature of word sense distributions. This paper presents a fully unsupervised topic modelling-based approach to sense frequency estimation, which is highly portable to different corpora and sense inventories, in being applicable to any part of speech, and not requiring a hierarchical sense inventory, parsing or parallel text. We demonstrate the effectiveness of the method over the tasks of predominant sense learning and sense distribution acquisition, and also the novel tasks of detecting senses which aren't attested in the corpus, and identifying novel senses in the corpus which aren't captured in the sense inventory.

1 Introduction

The automatic determination of word sense information has been a long-term pursuit of the NLP community (Agirre and Edmonds, 2006; Navigli, 2009). Word sense distributions tend to be Zipfian, and as such, a simple but surprisingly high-accuracy back-off heuristic for word sense disambiguation (WSD) is to tag each instance of a given word with its predominant sense (McCarthy et al., 2007). Such an approach requires knowledge of predominant senses; however, word sense distributions — and predominant senses too — vary from corpus to corpus. Therefore, methods for automatically learning predominant senses and sense distributions for specific corpora are required (Koeling et al., 2005; Lapata and Brew, 2004).

In this paper, we propose a method which uses topic models to estimate word sense distributions. This method is in principle applicable to all parts of speech, and moreover does not require a parser, a hierarchical sense representation or parallel text. Topic models have been used for WSD in a number of studies (Boyd-Graber et al., 2007; Li et al., 2010; Lau et al., 2012; Preiss and Stevenson, 2013; Cai et al., 2007; Knopp et al., 2013), but our work extends significantly on this earlier work in focusing on the acquisition of prior word sense distributions (and predominant senses).

Because of domain differences and the skewed nature of word sense distributions, it is often the case that some senses in a sense inventory will not be attested in a given corpus. A system capable of automatically finding such senses could reduce ambiguity, particularly in domain adaptation settings, while retaining rare but nevertheless viable senses. We further propose a method for applying our sense distribution acquisition system to the task of finding unattested senses — i.e., senses that are in the sense inventory but not attested in a given corpus. In contrast to the previous work of McCarthy et al. (2004a) on this topic which uses the sense ranking score from McCarthy et al. (2004b) to remove low-frequency senses from WordNet, we focus on finding senses that are unattested in the corpus on the premise that, given accurate disambiguation, rare senses in a corpus contribute to correct interpretation.

Corpus instances of a word can also correspond to senses that are not present in a given sense inventory. This can be due to, for example, words taking on new meanings over time (e.g. the rela-

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 259–270,
Baltimore, Maryland, USA, June 23-25 2014.

tively recent senses of *tablet* and *swipe* related to touchscreen computers) or domain-specific terms not being included in a more general-purpose sense inventory. A system for automatically identifying such novel senses — i.e. senses that are attested in the corpus but not in the sense inventory — would be a very valuable lexicographical tool for keeping sense inventories up-to-date (Cook et al., 2013). We further propose an application of our proposed method to the identification of such novel senses. In contrast to McCarthy et al. (2004b), the use of topic models makes this possible, using topics as a proxy for sense (Brody and Lapata, 2009; Yao and Durme, 2011; Lau et al., 2012). Earlier work on identifying novel senses focused on individual tokens (Erk, 2006), whereas our approach goes further in identifying groups of tokens exhibiting the same novel sense.

2 Background and Related Work

There has been a considerable amount of research on representing word senses and disambiguating usages of words in context (WSD) as, in order to produce computational systems that understand and produce natural language, it is essential to have a means of representing and disambiguating word sense. WSD algorithms require word sense information to disambiguate token instances of a given ambiguous word, e.g. in the form of sense definitions (Lesk, 1986), semantic relationships (Navigli and Velardi, 2005) or annotated data (Zhong and Ng, 2010). One extremely useful piece of information is the word sense prior or expected word sense frequency distribution. This is important because word sense distributions are typically skewed (Kilgarriff, 2004), and systems do far better when they take bias into account (Agirre and Martinez, 2004).

Typically, word frequency distributions are estimated with respect to a sense-tagged corpus such as SemCor (Miller et al., 1993), a 220,000 word corpus tagged with WordNet (Fellbaum, 1998) senses. Due to the expense of hand tagging, and sense distributions being sensitive to domain and genre, there has been some work on trying to estimate sense frequency information automatically (McCarthy et al., 2004b; Chan and Ng, 2005; Mohammad and Hirst, 2006; Chan and Ng, 2006). Much of this work has been focused on ranking word senses to find the predominant sense in a given corpus (McCarthy et al., 2004b; Mohammad and Hirst, 2006), which is a very powerful heuristic approach to WSD. Most WSD systems rely upon this heuristic for back-off in the absence of strong contextual evidence (McCarthy et al., 2007). McCarthy et al. (2004b) proposed a method which relies on distributionally similar words (nearest neighbours) associated with the target word in an automatically acquired thesaurus (Lin, 1998). The distributional similarity scores of the nearest neighbours are associated with the respective target word senses using a WordNet similarity measure, such as those proposed by Jiang and Conrath (1997) and Banerjee and Pedersen (2002). The word senses are ranked based on these similarity scores, and the most frequent sense is selected for the corpus that the distributional similarity thesaurus was trained over.

As well as sense ranking for predominant sense acquisition, automatic estimates of sense frequency distribution can be very useful for WSD for training data sampling purposes (Agirre and Martinez, 2004), entropy estimation (Jin et al., 2009), and prior probability estimates, all of which can be integrated within a WSD system (Chan and Ng, 2005; Chan and Ng, 2006; Lapata and Brew, 2004). Various approaches have been adopted, such as normalizing sense ranking scores to obtain a probability distribution (Jin et al., 2009), using subcategorisation information as an indication of verb sense (Lapata and Brew, 2004) or alternatively using parallel text (Chan and Ng, 2005; Chan and Ng, 2006; Agirre and Martinez, 2004).

The work of Boyd-Graber and Blei (2007) is highly related in that it extends the method of McCarthy et al. (2004b) to provide a generative model which assumes the words in a given document are generated according to the topic distribution appropriate for that document. They then predict the most likely sense for each word in the document based on the topic distribution and the words in context ("corroborators"), each of which, in turn, depends on the document's topic distribution. Using this approach, they get comparable results to McCarthy et al. when context is ignored (i.e. using a model with one topic), and at most a 1% improvement on SemCor when they use more topics in order to take context into account. Since the results do not improve on McCarthy et al. as regards sense distribution acquisition irrespective of context, we will compare our model with that proposed by McCarthy et al.

Recent work on finding novel senses has tended to focus on comparing diachronic corpora (Sagi et al., 2009; Cook and Stevenson, 2010; Gulordava and Baroni, 2011) and has also considered topic models (Lau et al., 2012). In a similar vein, Peirsman et al. (2010) considered the identification of words having a sense particular to one language variety with respect to another (specifically Belgian and Netherlandic Dutch). In contrast to these studies, we propose a model for comparing a corpus with a sense inventory. Carpuat et al. (2013) exploit parallel corpora to identify words in domain-specific monolingual corpora with previously-unseen translations; the method we propose does not require parallel data.

3 Methodology

Our methodology is based on the WSI system described in Lau et al. (2012),[1] which has been shown (Lau et al., 2012; Lau et al., 2013a; Lau et al., 2013b) to achieve state-of-the-art results over the WSI tasks from SemEval-2007 (Agirre and Soroa, 2007), SemEval-2010 (Manandhar et al., 2010) and SemEval-2013 (Navigli and Vannella, 2013; Jurgens and Klapaftis, 2013). The system is built around a Hierarchical Dirichlet Process (HDP: Teh et al. (2006)), a non-parametric variant of a Latent Dirichlet Allocation topic model (Blei et al., 2003) where the model automatically optimises the number of topics in a fully-unsupervised fashion over the training data.

To learn the senses of a target lemma, we train a single topic model per target lemma. The system reads in a collection of usages of that lemma, and automatically induces topics (= senses) in the form of a multinomial distribution over words, and per-usage topic assignments (= probabilistic sense assignments) in the form of a multinomial distribution over topics. Following Lau et al. (2012), we assign one topic to each usage by selecting the topic that has the highest cumulative probability density, based on the topic allocations of all words in the context window for that usage.[2] Note that in their original work, Lau et al. (2012) experimented with the use of features extracted from a dependency parser. Due to the computational overhead associated with these features, and the fact that the empirical impact of the features was found to be marginal, we make no use of parser-based features in this paper.[3]

The induced topics take the form of word multinomials, and are often represented by the top-N words in descending order of conditional probability. We interpret each topic as a sense of the target lemma.[4] To illustrate this, we give the example of topics induced by the HDP model for *network* in Table 1.

We refer to this method as HDP-WSI henceforth.[5]

In predominant sense acquisition, the task is to learn, for each target lemma, the most frequently occurring word sense in a particular domain or corpus, relative to a predefined sense inventory. The WSI system provides us with a topic allocation per usage of a given word, from which we can derive a distribution of topics over usages and a **predominant topic**. In order to map this onto the **predominant sense**, we need to have some way of aligning a topic with a sense. We design our topic–sense alignment methodology with portability in mind — it should be applicable to any sense inventory. As such, our alignment methodology assumes only that we have access to a conventional sense gloss or definition for each sense, and does not rely on ontological/structural knowledge (e.g. the WordNet hierarchy).

To compute the similarity between a sense and a topic, we first convert the words in the gloss/definition into a multinomial distribution over words, based on simple maximum likelihood estimation.[6] We then calculate the Jensen–Shannon divergence between the multinomial distribution (over words) of the gloss and that of the topic, and convert the divergence value into a similarity score by subtracting it from 1. Formally, the similarity sense s_i and topic t_j is:

$$\text{sim}(s_i, t_j) = 1 - \text{JS}(S \| T) \quad (1)$$

where S and T are the multinomial distributions

[1] Based on the implementation available at: `https://github.com/jhlau/hdp-wsi`

[2] This includes all words in the usage sentence except stopwords, which were filtered in the preprocessing step.

[3] For hyper-parameters α and γ, we used 0.1 for both. We did not tune the parameters, and opted to use the default parameters introduced in Teh et al. (2006).

[4] To avoid confusion, we will refer to the HDP-induced topics as *topics*, and reserve the term *sense* to denote senses in a sense inventory.

[5] The code used to learn predominant sense and run all experiments described in this paper is available at: `https://github.com/jhlau/predom_sense`.

[6] Words are tokenised using OpenNLP and lemmatised with Morpha (Minnen et al., 2001). We additionally remove the target lemma, stopwords and words that are less than 3 characters in length.

Topic Num	Top-10 Terms
1	network support @card@ information research service group development community member
2	service @card@ road company transport rail area government network public
3	network social model system family structure analysis form relationship neural
4	network @card@ computer system service user access internet datum server
5	system network management software support corp company service application product
6	@card@ radio news television show bbc programme call think film
7	police drug criminal terrorist intelligence network vodafone iraq attack cell
8	network atm manager performance craigavon group conference working modelling assistant
9	root panos comenius etd unipalm lse brazil telephone xxx discuss

Table 1: An example to illustrate the topics induced for *network* by the HDP model. The top-10 highest probability terms are displayed to represent each topic (*@card@* denotes a tokenised cardinal number).

over words for sense s_i and topic t_j, respectively, and $\text{JS}(X\|Y)$ is the Jensen–Shannon divergence for distribution X and Y.

To learn the predominant sense, we compute the **prevalence score** of each sense and take the sense with the highest prevalence score as the predominant sense. The prevalence score for a sense is computed by summing the product of its similarity scores with each topic (i.e. $\text{sim}(s_i, t_j)$) and the prior probability of the topic in question (based on maximum likelihood estimation). Formally, the prevalence score of sense s_i is given as follows:

$$\text{prevalence}(s_i) = \sum_{j}^{T} (\text{sim}(s_i, t_j) \times P(t_j)) \quad (2)$$
$$= \sum_{j}^{T} \left(\text{sim}(s_i, t_j) \times \frac{f(t_j)}{\sum_{k}^{T} f(t_k)} \right)$$

where $f(t_j)$ is the frequency of topic t_j (i.e. the number of usages assigned to topic t_j), and T is the number of topics.

The intuition behind the approach is that the predominant sense should be the sense that has relatively high similarity (in terms of lexical overlap) with high-probability topic(s).

4 WordNet Experiments

We first test the proposed method over the tasks of predominant sense learning and sense distribution induction, using the WordNet-tagged dataset of Koeling et al. (2005), which is made up of 3 collections of documents: a domain-neutral corpus (BNC), and two domain-specific corpora (SPORTS and FINANCE). For each domain, annotators were asked to sense-annotate a random selection of sentences for each of 40 target nouns, based on WordNet v1.7. The predominant sense and distribution across senses for each target lemma was obtained by aggregating over the sense annotations. The authors evaluated their method in terms of WSD accuracy over a given corpus, based on assigning all instances of a target word with the predominant sense learned from that corpus. For the remainder of the paper, we denote their system as MKWC.

To compare our system (HDP-WSI) with MKWC, we apply it to the three datasets of Koeling et al. (2005). For each dataset, we use HDP to induce topics for each target lemma, compute the similarity between the topics and the WordNet senses (Equation (1)), and rank the senses based on the prevalence scores (Equation (2)). In addition to the WSD accuracy based on the predominant sense inferred from a particular corpus, we additionally compute: (1) Acc_{UB}, the upper bound for the first sense-based WSD accuracy (using the gold standard predominant sense for disambiguation);[7] and (2) ERR, the error rate reduction between the accuracy for a given system (Acc) and the upper bound (Acc_{UB}), calculated as follows:

$$\text{ERR} = 1 - \frac{\text{Acc}_{\text{UB}} - \text{Acc}}{\text{Acc}_{\text{UB}}}$$

Looking at the results in Table 2, we see little difference in the results for the two methods, with MKWC performing better over two of the datasets (BNC and SPORTS) and HDP-WSI performing better over the third (FINANCE), but all differences are small. Based on the McNemar's Test with Yates correction for continuity, MKWC is significantly better over BNC and HDP-WSI is significantly better over FINANCE ($p < 0.0001$ in both cases), but the difference over SPORTS is not statistically significance ($p > 0.1$). Note that there is still much room for improvement with

[7] The upper bound for a WSD approach which tags all token occurrences of a given word with the same sense, as a first step towards context-sensitive unsupervised WSD.

Dataset	FS_{CORPUS} Acc_{UB}	MKWC Acc	MKWC ERR	HDP-WSI Acc	HDP-WSI ERR
BNC	0.524	**0.407**	(0.777)	0.376	(0.718)
FINANCE	0.801	0.499	(0.623)	**0.555**	(0.693)
SPORTS	0.774	**0.437**	(0.565)	0.422	(0.545)

Table 2: WSD accuracy for MKWC and HDP-WSI on the WordNet-annotated datasets, as compared to the upper-bound based on actual first sense in the corpus (higher values indicate better performance; the **best** system in each row [other than the FS_{CORPUS} upper bound] is indicated in boldface).

Dataset	MKWC	HDP-WSI
BNC	0.226	**0.214**
FINANCE	0.426	**0.375**
SPORTS	0.420	**0.363**

Table 3: Sense distribution evaluation of MKWC and HDP-WSI on the WordNet-annotated datasets, evaluated using JS divergence (lower values indicate better performance; the **best** system in each row is indicated in boldface).

both systems, as we see in the gap between the upper bound (based on perfect determination of the first sense) and the respective system accuracies.

Given that both systems compute a continuous-valued prevalence score for each sense of a target lemma, a distribution of senses can be obtained by normalising the prevalence scores across all senses. The predominant sense learning task of McCarthy et al. (2007) evaluates the ability of a method to identify only the head of this distribution, but it is also important to evaluate the full sense distribution (Jin et al., 2009). To this end, we introduce a second evaluation metric: the Jensen–Shannon (JS) divergence between the inferred sense distribution and the gold-standard sense distribution, noting that smaller values are better in this case, and that it is now theoretically possible to obtain a JS divergence of 0 in the case of a perfect estimate of the sense distribution. Results are presented in Table 3.

HDP-WSI consistently achieves lower JS divergence, indicating that the distribution of senses that it finds is closer to the gold standard distribution. Testing for statistical significance over the paired JS divergence values for each lemma using the Wilcoxon signed-rank test, the result for FINANCE is significant ($p < 0.05$) but the results for the other two datasets are not ($p > 0.1$ in each case).

Dataset	FS_{CORPUS} Acc_{UB}	FS_{DICT} Acc	FS_{DICT} ERR	HDP-WSI Acc	HDP-WSI ERR
UKWAC	0.574	0.387	(0.674)	**0.514**	(0.895)
TWITTER	0.468	0.297	(0.635)	**0.335**	(0.716)

Table 4: WSD accuracy for HDP-WSI on the Macmillan-annotated datasets, as compared to the upper-bound based on actual first sense in the corpus (higher values indicate better performance; the **best** system in each row [other than the FS_{CORPUS} upper bound] is indicated in boldface).

Dataset	FS_{CORPUS}	FS_{DICT}	HDP-WSI
UKWAC	0.210	0.393	**0.156**
TWITTER	0.259	0.472	**0.171**

Table 5: Sense distribution evaluation of HDP-WSI on the Macmillan-annotated datasets as compared to corpus- and dictionary-based first sense methods, evaluated using JS divergence (lower values indicate better performance; the **best** system in each row is indicated in boldface).

To summarise, the results for MKWC and HDP-WSI are fairly even for predominant sense learning (each outperforms the other at a level of statistical significance over one dataset), but HDP-WSI is better at inducing the overall sense distribution.

It is important to bear in mind that MKWC in these experiments makes use of full-text parsing in calculating the distributional similarity thesaurus, and the WordNet graph structure in calculating the similarity between associated words and different senses. Our method, on the other hand, uses no parsing, and only the synset definitions (and not the graph structure) of WordNet.[8] The non-reliance on parsing is significant in terms of portability to text sources which are less amenable to parsing (such as Twitter: (Baldwin et al., 2013)), and the non-reliance on the graph structure of WordNet is significant in terms of portability to conventional "flat" sense inventories. While comparable results on a different dataset have been achieved with a proximity thesaurus (McCarthy et al., 2007) compared to a dependency one,[9] it is not stated how

[8] McCarthy et al. (2004b) obtained good results with definition overlap, but their implementation uses the relation structure alongside the definitions (Banerjee and Pedersen, 2002). Iida et al. (2008) demonstrate that further extensions using distributional data are required when applying the method to resources without hierarchical relations.

[9] The thesauri used in the reimplementation of MKWC in this paper were obtained from `http://webdocs.cs.ualberta.ca/~lindek/downloads.htm`.

wide a window is needed for the proximity thesaurus. This could be a significant issue with Twitter data, where context tends to be limited. In the next section, we demonstrate the robustness of the method in experimenting with two new datasets, based on Twitter and a web corpus, and the Macmillan English Dictionary.

5 Macmillan Experiments

In our second set of experiments, we move to a new dataset (Gella et al., to appear) based on text from ukWaC (Ferraresi et al., 2008) and Twitter, and annotated using the Macmillan English Dictionary[10] (henceforth "Macmillan"). For the purposes of this research, the choice of Macmillan is significant in that it is a conventional dictionary with sense definitions and examples, but no linking between senses.[11] In terms of the original research which gave rise to the sense-tagged dataset, Macmillan was chosen over WordNet for reasons including: (1) the well-documented difficulties of sense tagging with fine-grained WordNet senses (Palmer et al., 2004; Navigli et al., 2007); (2) the regular update cycle of Macmillan (meaning it contains many recently-emerged senses); and (3) the finding in a preliminary sense-tagging task that it better captured Twitter usages than WordNet (and also OntoNotes: Hovy et al. (2006)).

The dataset is made up of 20 target nouns which were selected to span the high- to mid-frequency range in both Twitter and the ukWaC corpus, and have at least 3 Macmillan senses. The average sense ambiguity of the 20 target nouns in Macmillan is 5.6 (but 12.3 in WordNet). 100 usages of each target noun were sampled from each of Twitter (from a crawl over the time period Jan 3–Feb 28, 2013 using the Twitter Streaming API) and ukWaC, after language identification using `langid.py` (Lui and Baldwin, 2012) and POS tagging (based on the CMU ARK Twitter POS tagger v2.0 (Owoputi et al., 2012) for Twitter, and the POS tags provided with the corpus for ukWaC). Amazon Mechanical Turk (AMT) was then used to 5-way sense-tag each usage relative to Macmillan, including allowing the annotators the option to label a usage as "Other" in instances where the usage was not captured by any of the Macmillan senses. After quality control over the annotators/annotations (see Gella et al. (to appear) for details), and aggregation of the annotations into a single sense per usage (possibly "Other"), there were 2000 sense-tagged ukWaC sentences and Twitter messages over the 20 target nouns. We refer to these two datasets as UKWAC and TWITTER henceforth.

To apply our method to the two datasets, we use HDP-WSI to train a model for each target noun, based on the combined set of usages of that lemma in each of the two background corpora, namely the original Twitter crawl that gave rise to the TWITTER dataset, and all of ukWaC.

5.1 Learning Sense Distributions

As in Section 4, we evaluate in terms of WSD accuracy (Table 4) and JS divergence over the gold-standard sense distribution (Table 5). We also present the results for: (a) a supervised baseline ("FS_{CORPUS}"), based on the most frequent sense in the corpus; and (b) an unsupervised baseline ("FS_{DICT}"), based on the first-listed sense in Macmillan. In each case, the sense distribution is based on allocating all probability mass for a given word to the single sense identified by the respective method.

We first notice that, despite the coarser-grained senses of Macmillan as compared to WordNet, the upper bound WSD accuracy using Macmillan is comparable to that of the WordNet-based datasets over the balanced BNC, and quite a bit lower than that of the two domain corpora of Koeling et al. (2005). This suggests that both datasets are diverse in domain and content.

In terms of WSD accuracy, the results over UKWAC (ERR = 0.895) are substantially higher than those for BNC, while those over TWITTER (ERR = 0.716) are comparable. The accuracy is significantly higher than the dictionary-based first sense baseline (FS_{DICT}) over both datasets (McNemar's test; $p < 0.0001$), and the ERR is also considerably higher than for the two domain datasets in Section 4 (FINANCE and SPORTS). One cause of difficulty in sense-modelling TWITTER is large numbers of missing senses, with 12.3% of usages in TWITTER and 6.6% in UKWAC having no corresponding Macmillan sense.[12] This challenges the assumption built into the sense prevalence calculation that all topics will align to a pre-existing sense, a point we return to in Section 5.2.

[10] `http://www.macmillandictionary.com/`

[11] Strictly speaking, there is limited linking in the form of sets of synonyms in Macmillan, but we choose to not use this information in our research.

[12] The relative occurrence of unlisted/unclear senses in the datasets of Koeling et al. (2005) is comparable to UKWAC.

Dataset	P	R	F
UKWAC	0.73	0.85	0.74
TWITTER	0.56	0.88	0.65

Table 6: Evaluation of our method for identifying unattested senses, averaged over 10 runs of 10-fold cross validation

The JS divergence results for both datasets are well below (= better than) the results for all three WordNet-based datasets, and also superior to both the supervised and unsupervised first-sense baselines. Part of the reason for this improvement is simply that the average polysemy in Macmillan (5.6 senses per target lemma) is slightly less than in WordNet (6.7 senses per target lemma),[13] making the task slightly easier in the Macmillan case.

5.2 Identification of Unattested Senses

We observed in Section 5.1 that there are relatively frequent occurrences of usages (e.g. 12.3% for TWITTER) which aren't captured by Macmillan. Conversely, there are also senses in Macmillan which aren't attested in the annotated sample of usages. Specifically, of the 112 senses defined for the 20 target lemmas, 25 (= 22.3%) of the senses are not attested in the 2000 usages in either corpora. Given that our methodology computes a prevalence score for each sense, it can equally be applied to the detection of these unattested senses, and it is this task that we address in this section: the identification of senses that are defined in the sense inventory but not attested in a given corpus.

Intuitively, an unused sense should have low similarity with the HDP induced topics. As such, we introduce sense-to-topic affinity, a measure that estimates how likely a sense is not attested in the corpus:

$$\text{st-affinity}(s_i) = \frac{\sum_j^T \text{sim}(s_i, t_j)}{\sum_k^S \sum_l^T \text{sim}(s_k, t_l)} \quad (3)$$

where $\text{sim}(s_i, t_j)$ is carried over from Equation (1), and T and S represent the number of topics and senses, respectively.

We treat the task of identification of unused senses as a binary classification problem, where the goal is to find a sense-to-topic affinity threshold below which a sense will be considered to be unused. We pool together all the senses and run 10-fold cross validation to learn the threshold for identifying unused senses,[14] evaluated using sense-level precision (P), recall (R) and F-score (F) at detecting unattested senses. We repeat the experiment 10 times (partitioning the items randomly into folds) and collect the mean precision, recall and F-scores across the 10 runs. We found encouraging results for the task, as detailed in Table 6. For the threshold, the average value with standard deviation is 0.092 ± 0.044 over UKWAC and 0.125 ± 0.052 over TWITTER, indicating relative stability in the value of the threshold both internally within a dataset, and also across datasets.

5.3 Identification of Novel Senses

In both TWITTER and UKWAC, we observed frequent occurrences of usages of our target nouns which didn't map onto a pre-existing Macmillan sense. A natural question to ask is whether our method can be used to predict word senses that are missing from our sense inventory, and identify usages associated with each such missing sense. We will term these "novel senses", and define "novel sense identification" to be the task of identifying new senses that are not recorded in the inventory but are seen in the corpus.

An immediate complication in evaluating novel sense identification is that we are attempting to identify senses which explicitly aren't in our sense inventory. This contrasts with the identification of unattested senses, e.g., where we were attempting to identify which of the *known* senses wasn't observed in the corpus. Also, while we have annotations of "Other" usages in TWITTER and UKWAC, there is no real expectation that all such usages will correspond to the same sense: in practice, they are attributable to a myriad of effects such as incorporation in a non-compositional multiword expression, and errors in POS tagging (i.e. the usage not being nominal). As such, we can't use the "Other" annotations to evaluate novel sense identification. The evaluation of systems for this task is a known challenge, which we address similarly to Erk (2006) by artificially synthesising novel senses through removal of senses from the sense inventory. In this way, even if we remove multiple senses for a given word, we still have access to information about which usages correspond to

[13] Note that the set of lemmas differs between the respective datasets, so this isn't an accurate reflection of the relative granularity of the two dictionaries.

[14] We used a fixed step and increment at steps of 0.001, up to the max value of st-affinity when optimising the threshold.

No. Lemmas with a Removed Sense	Relative Freq of Removed Sense	Threshold Mean±stdev	P	R	F
20	0.0–0.2	0.052±0.009	0.35	0.42	0.36
9	0.2–0.4	0.089±0.024	0.24	0.59	0.29
6	0.4–0.6	0.061±0.004	0.63	0.64	0.63

Table 7: Classification of usages with novel sense for all target lemmas.

No. Lemmas with a Removed Sense	Relative Freq of Removed Sense	Threshold Mean±stdev	P	R	F
9	0.2–0.4	0.093±0.023	0.50	0.66	0.52
6	0.4–0.6	0.099±0.018	0.73	0.90	0.80

Table 8: Classification of usages with novel sense for target lemmas with a removed sense.

which novel sense. An additional advantage of this procedure is that it allows us to control an important property of novel senses: their frequency of occurrence.

In the experiments that follow, we randomly select senses for removal from three frequency bands: low, medium and high frequency senses. Frequency is defined by relative occurrence in the annotated usages: low = 0.0–0.2; medium = 0.2–0.4; and high = 0.4–0.6. Note that we do not consider high-frequency senses with frequency higher than 0.6, as it is rare for a medium- to high-frequency word to take on a novel sense which is then the predominant sense in a given corpus. Note also that not all target lemmas will have a novel sense through synthesis, as they may have no senses that fall within the indicated bounds of relative occurrence (e.g. if $> 60\%$ of usages are a single sense). For example, only 6 of our 20 target nouns have senses which are candidates for high-frequency novel senses.

As before, we treat the novel sense identification task as a classification problem, although with a significantly different formulation: we are no longer attempting to identify pre-existing senses, as novel senses are by definition not included in the sense inventory. Instead, we are seeking to identify clusters of usages which are instances of a novel sense, e.g. for presentation to a lexicographer as part of a dictionary update process (Rundell and Kilgarriff, 2011; Cook et al., 2013). That is, for each usage, we want to classify whether it is an instance of a given novel sense.

A usage that corresponds to a novel sense should have a topic that does not align well with any of the pre-existing senses in the sense inventory. Based on this intuition, we introduce topic-to-sense affinity to estimate the similarity of a topic to the set of senses, as follows:

$$\text{ts-affinity}(t_j) = \frac{\sum_i^S \text{sim}(s_i, t_j)}{\sum_l^T \sum_k^S \text{sim}(s_k, t_l)} \quad (4)$$

where, once again, $\text{sim}(s_i, t_j)$ is defined as in Equation (1), and T and S represent the number of topics and senses, respectively.

Using topic-to-sense affinity as the sole feature, we pool together all instances and optimise the affinity feature to classify instances that have novel senses. Evaluation is done by computing the mean precision, recall and F-score across 10 separate runs; results are summarised in Table 7. Note that we evaluate only over UKWAC in this section, for ease of presentation.

The results show that instances with high-frequency novel senses are more easily identifiable than instances with medium/low-frequency novel senses. This is unsurprising given that high-frequency senses have a higher probability of generating related topics (sense-related words are observed more frequently in the corpus), and as such are more easily identifiable.

We are interested in understanding whether pooling all instances — instances from target lemmas that have a sense artificially removed and those that do not — impacted the results (recall that not all target lemmas have a removed sense). To that end, we chose to include only instances from lemmas with a removed sense, and repeated the experiment for the medium- and high-frequency novel sense condition (for the low-frequency condition, all target lemmas have a novel sense). In other words, we are assuming knowledge of which words have novel sense, and the task is to identify specifically what the novel sense is, as represented by novel usages. Results are presented in Table 8.

No. of Lemmas with a Removed Sense	No. of Lemmas without a Removed Sense	Relative Freq of Removed Sense	Wilcoxon Rank Sum p-value
10	0	0.0–0.2	0.4543
9	11	0.2–0.4	0.0391
6	14	0.4–0.6	0.0247

Table 9: Wilcoxon Rank Sum p-value results for testing target lemmas with removed sense vs. target lemmas without removed sense using novelty.

From the results, we see that the F-scores improved notably. This reveals that an additional step is necessary to determine whether a target lemma has a potential novel sense before feeding its instances to learn which of them contains the usage of the novel sense.

In the last experiment, we propose a new measure to tackle this: the identification of target lemmas that have a novel sense. We introduce novelty, a measure of the likelihood of a target lemma w having a novel sense:

$$\text{novelty}(w) = \min_{t_j} \left(\max_{s_i} \frac{\text{sim}(s_i, t_j)}{f(t_j)} \right) \quad (5)$$

where $f(t_j)$ is the frequency of topic t_j in the corpus. The intuition behind novelty is that a target lemma with a novel sense should have a (somewhat-)frequent topic that has low association with any sense. That we use the frequency rather than the probability of the topic here is deliberate, as topics with a higher raw number of occurrences (whether as a low-probability topic for a high-frequency word, or a high-probability topic for a low-frequency word) are indicative of a novel word sense.

For each of our three datasets (with low-, medium- and high-frequency novel senses, respectively), we compute the novelty of the target lemmas and the p-value of a one-tailed Wilcoxon rank sum test to test if the two groups of lemmas (i.e. lemmas with a novel sense vs. lemmas without a novel sense) are statistically different.[15] Results are presented in Table 9. We see that the novelty measure can readily identify target lemmas with high- and medium-frequency novel senses ($p < 0.05$), but the results are less promising for the low-frequency novel senses.

6 Discussion

Our methodologies for the two proposed tasks of identifying unused and novel senses are simple extensions to demonstrate the flexibility and robustness of our methodology. Future work could pursue a more sophisticated methodology, using non-linear combinations of $\text{sim}(s_i, t_j)$ for computing the affinity measures or multiple features in a supervised context. We contend, however, that these extensions are ultimately a preliminary demonstration to the flexibility and robustness of our methodology.

A natural next step for this research would be to couple sense distribution estimation and the detection of unattested senses with evidence from the context, using topics or other information about the local context (e.g. Agirre and Soroa (2009)) to carry out unsupervised WSD of individual token occurrences of a given word.

In summary, we have proposed a topic modelling-based method for estimating word sense distributions, based on Hierarchical Dirichlet Processes and the earlier work of Lau et al. (2012) on word sense induction, in probabilistically mapping the automatically-learned topics to senses in a sense inventory. We evaluated the ability of the method to learn predominant senses and induce word sense distributions, based on a broad range of datasets and two separate sense inventories. In doing so, we established that our method is comparable to the approach of McCarthy et al. (2007) at predominant sense learning, and superior at inducing word sense distributions. We further demonstrated the applicability of the method to the novel tasks of detecting word senses which are unattested in a corpus, and identifying novel senses which are found in a corpus but not captured in a word sense inventory.

Acknowledgements

We wish to thank the anonymous reviewers for their valuable comments. This research was supported in part by funding from the Australian Research Council.

[15] Note that the number of words with low-frequency novel senses here is restricted to 10 (cf. 20 in Table 7) to ensure we have both positive and negative lemmas in the dataset.

References

Eneko Agirre and Philip Edmonds, editors. 2006. *Word Sense Disambiguation: Algorithms and Applications*. Springer, Dordrecht, Netherlands.

Eneko Agirre and David Martinez. 2004. Unsupervised WSD based on automatically retrieved examples: The importance of bias. In *Proceedings of EMNLP 2004*, pages 25–32, Barcelona, Spain.

Eneko Agirre and Aitor Soroa. 2007. SemEval-2007 task 02: Evaluating word sense induction and discrimination systems. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 7–12, Prague, Czech Republic.

Eneko Agirre and Aitor Soroa. 2009. Personalizing PageRank for word sense disambiguation. In *Proceedings of the 12th Conference of the EACL (EACL 2009)*, pages 33–41, Athens, Greece.

Timothy Baldwin, Paul Cook, Marco Lui, Andrew MacKinlay, and Li Wang. 2013. How noisy social media text, how diffrnt social media sources? In *Proceedings of the 6th International Joint Conference on Natural Language Processing (IJCNLP 2013)*, pages 356–364, Nagoya, Japan.

Satanjeev Banerjee and Ted Pedersen. 2002. An adapted Lesk algorithm for word sense disambiguation using WordNet. In *Proceedings of the 3rd International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2002)*, pages 136–145, Mexico City, Mexico.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Jordan Boyd-Graber and David Blei. 2007. Putop: Turning predominant senses into a topic model for word sense disambiguation. In *Proc. of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 277–281, Prague, Czech Republic.

Jordan Boyd-Graber, David Blei, and Xiaojin Zhu. 2007. A topic model for word sense disambiguation. In *Proc. of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1024–1033, Prague, Czech Republic.

Samuel Brody and Mirella Lapata. 2009. Bayesian word sense induction. In *Proceedings of the 12th Conference of the EACL (EACL 2009)*, pages 103–111, Athens, Greece.

Jun Fu Cai, Wee Sun Lee, and Yee Whye Teh. 2007. NUS-ML: Improving word sense disambiguation using topic features. In *Proc. of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 249–252, Prague, Czech Republic.

Marine Carpuat, Hal Daumé III, Katharine Henry, Ann Irvine, Jagadeesh Jagarlamudi, and Rachel Rudinger. 2013. SenseSpotting: Never let your parallel data tie you to an old domain. In *Proc. of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pages 1435–1445, Sofia, Bulgaria.

Yee Seng Chan and Hwee Tou Ng. 2005. Word sense disambiguation with distribution estimation. In *Proc. of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*, pages 1010–1015, Edinburgh, UK.

Yee Seng Chan and Hwee Tou Ng. 2006. Estimating class priors in domain adaptation for word sense disambiguation. In *Proc. of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 89–96, Sydney, Australia.

Paul Cook and Suzanne Stevenson. 2010. Automatically identifying changes in the semantic orientation of words. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC 2010)*, pages 28–34, Valletta, Malta.

Paul Cook, Jey Han Lau, Michael Rundell, Diana McCarthy, and Timothy Baldwin. 2013. A lexicographic appraisal of an automatic approach for detecting new word senses. In *Proceedings of eLex 2013*, pages 49–65, Tallinn, Estonia.

Katrin Erk. 2006. Unknown word sense detection as outlier detection. In *Proc. of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 128–135, New York City, USA.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, USA.

Adriano Ferraresi, Eros Zanchetta, Marco Baroni, and Silvia Bernardini. 2008. Introducing and evaluating ukWaC, a very large web-derived corpus of English. In *Proc. of the 4th Web as Corpus Workshop: Can we beat Google*, pages 47–54, Marrakech, Morocco.

Spandana Gella, Paul Cook, and Timothy Baldwin. to appear. One sense per tweeter ... and other lexical semantic tales of Twitter. In *Proceedings of the 14th Conference of the EACL (EACL 2014)*, Gothenburg, Sweden.

Kristina Gulordava and Marco Baroni. 2011. A distributional similarity approach to the detection of semantic change in the Google Books Ngram corpus. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 67–71, Edinburgh, UK.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: The 90% solution. In *Proceedings of*

the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, pages 57–60, New York City, USA.

Ryu Iida, Diana McCarthy, and Rob Koeling. 2008. Gloss-based semantic similarity metrics for predominant sense acquisition. In *Proc. of the Third International Joint Conference on Natural Language Processing*, pages 561–568.

Jay Jiang and David Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings on International Conference on Research in Computational Linguistics*, pages 19–33, Taipei, Taiwan.

Peng Jin, Diana McCarthy, Rob Koeling, and John Carroll. 2009. Estimating and exploiting the entropy of sense distributions. In *Proceedings of the North American Chapter of the Association for Computational Linguistics – Human Language Technologies 2009 (NAACL HLT 2009): Short Papers*, pages 233–236, Boulder, USA.

David Jurgens and Ioannis Klapaftis. 2013. Semeval-2013 task 13: Word sense induction for graded and non-graded senses. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval 2013)*, pages 290–299, Atlanta, USA.

Adam Kilgarriff. 2004. How dominant is the commonest sense of a word? Technical Report ITRI-04-10, Information Technology Research Institute, University of Brighton.

Johannes Knopp, Johanna Völker, and Simone Paolo Ponzetto. 2013. Topic modeling for word sense induction. In *Proc. of the International Conference of the German Society for Computational Linguistics and Language Technology*, pages 97–103, Darmstadt, Germany.

Rob Koeling, Diana McCarthy, and John Carroll. 2005. Domain-specific sense distributions and predominant sense acquisition. In *Proceedings of the 2005 Conference on Empirical Methods in Natural Language Processing (EMNLP 2005)*, pages 419–426, Vancouver, Canada.

Mirella Lapata and Chris Brew. 2004. Verb class disambiguation using informative priors. *Computational Linguistics*, 30(1):45–75.

Jey Han Lau, Paul Cook, Diana McCarthy, David Newman, and Timothy Baldwin. 2012. Word sense induction for novel sense detection. In *Proceedings of the 13th Conference of the EACL (EACL 2012)*, pages 591–601, Avignon, France.

Jey Han Lau, Paul Cook, and Timothy Baldwin. 2013a. unimelb: Topic modelling-based word sense induction. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval 2013)*, pages 307–311, Atlanta, USA.

Jey Han Lau, Paul Cook, and Timothy Baldwin. 2013b. unimelb: Topic modelling-based word sense induction for web snippet clustering. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval 2013)*, pages 217–221, Atlanta, USA.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 1986 SIGDOC Conference*, pages 24–26, Ontario, Canada.

Linlin Li, Benjamin Roth, and Caroline Sporleder. 2010. Topic models for word sense disambiguation and token-based idiom detection. In *Proc. of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1138–1147, Uppsala, Sweden.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 36th Annual Meeting of the ACL and 17th International Conference on Computational Linguistics (COLING/ACL-98)*, pages 768–774, Montreal, Canada.

Marco Lui and Timothy Baldwin. 2012. langid.py: An off-the-shelf language identification tool. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012) Demo Session*, pages 25–30, Jeju, Republic of Korea.

Suresh Manandhar, Ioannis Klapaftis, Dmitriy Dligach, and Sameer Pradhan. 2010. SemEval-2010 Task 14: Word sense induction & disambiguation. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 63–68, Uppsala, Sweden.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2004a. Automatic identification of infrequent word senses. In *Proc. of the 20th International Conference of Computational Linguistics, COLING-2004*, pages 1220–1226, Geneva, Switzerland.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2004b. Finding predominant senses in untagged text. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004)*, pages 280–287, Barcelona, Spain.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2007. Unsupervised acquisition of predominant word senses. *Computational Linguistics*, 4(33):553–590.

George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. 1993. A semantic concordance. In *Proc. of the ARPA Workshop on Human Language Technology*, pages 303–308.

Guido Minnen, John Carroll, and Darren Pearce. 2001. Applied morphological processing of English. *Natural Language Engineering*, 7(3):207–223.

Saif Mohammad and Graeme Hirst. 2006. Determining word sense dominance using a thesaurus. In *Proc. of EACL-2006*, pages 121–128, Trento, Italy.

Roberto Navigli and Daniele Vannella. 2013. SemEval-2013 task 11: Word sense induction and disambiguation within an end-user application. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval 2013)*, pages 193–201, Atlanta, USA.

Roberto Navigli and Paola Velardi. 2005. Structural semantic interconnections: a knowledge-based approach to word sense disambiguation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1075–1088.

Roberto Navigli, Kenneth C. Litkowski, and Orin Hargraves. 2007. SemEval-2007 task 07: Coarse-grained English all-words task. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 30–35, Prague, Czech Republic.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys*, 41(2).

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, and Nathan Schneider. 2012. Part-of-speech tagging for Twitter: Word clusters and other advances. Technical Report CMU-ML-12-107, Machine Learning Department, Carnegie Mellon University.

Martha Palmer, Olga Babko-Malaya, and Hoa Trang Dang. 2004. Different sense granularities for different applications. In *Proceedings of the HLT-NAACL 2004 Workshop: 2nd Workshop on Scalable Natural Language Understanding*, pages 49–56, Boston, USA.

Yves Peirsman, Dirk Geeraerts, and Dirk Speelman. 2010. The automatic identification of lexical variation between language varieties. *Natural Language Engineering*, 16(4):469–491.

Judita Preiss and Mark Stevenson. 2013. Unsupervised domain tuning to improve word sense disambiguation. In *Proc. of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 680–684, Atlanta, USA.

Michael Rundell and Adam Kilgarriff. 2011. Automating the creation of dictionaries: where will it all end? In Fanny Meunier, Sylvie De Cock, Gaëtanelle Gilquin, and Magali Paquot, editors, *A Taste for Corpora. In honour of Sylviane Granger*, pages 257–282. John Benjamins, Amsterdam, Netherlands.

Eyal Sagi, Stefan Kaufmann, and Brady Clark. 2009. Semantic density analysis: Comparing word meaning across time and space. In *Proceedings of the EACL 2009 Workshop on GEMS: GEometrical Models of Natural Language Semantics*, pages 104–111, Athens, Greece.

Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. 2006. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101:1566–1581.

Xuchen Yao and Benjamin Van Durme. 2011. Nonparametric Bayesian word sense induction. In *Proceedings of TextGraphs-6: Graph-based Methods for Natural Language Processing*, pages 10–14, Portland, USA.

Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proc. of the ACL 2010 System Demonstrations*, pages 78–83, Uppsala, Sweden.

Learning to Automatically Solve Algebra Word Problems

Nate Kushman[†], **Yoav Artzi**[‡], **Luke Zettlemoyer**[‡], **and Regina Barzilay**[†]

[†] Computer Science and Articial Intelligence Laboratory, Massachusetts Institute of Technology
{nkushman, regina}@csail.mit.edu
[‡] Computer Science & Engineering, University of Washington
{yoav, lsz}@cs.washington.edu

Abstract

We present an approach for automatically learning to solve algebra word problems. Our algorithm reasons across sentence boundaries to construct and solve a system of linear equations, while simultaneously recovering an alignment of the variables and numbers in these equations to the problem text. The learning algorithm uses varied supervision, including either full equations or just the final answers. We evaluate performance on a newly gathered corpus of algebra word problems, demonstrating that the system can correctly answer almost 70% of the questions in the dataset. This is, to our knowledge, the first learning result for this task.

1 Introduction

Algebra word problems concisely describe a world state and pose questions about it. The described state can be modeled with a system of equations whose solution specifies the questions' answers. For example, Figure 1 shows one such problem. The reader is asked to infer how many children and adults were admitted to an amusement park, based on constraints provided by ticket prices and overall sales. This paper studies the task of learning to automatically solve such problems given only the natural language.[1]

Solving these problems requires reasoning across sentence boundaries to find a system of equations that concisely models the described semantic relationships. For example, in Figure 1, the total ticket revenue computation in the second equation summarizes facts about ticket prices and total sales described in the second, third, and fifth sentences. Furthermore, the first equation models an implicit semantic relationship, namely that the children and adults admitted are non-intersecting subsets of the set of people who entered the park.

Word problem
An amusement park sells 2 kinds of tickets. Tickets for children cost $1.50. Adult tickets cost $4. On a certain day, 278 people entered the park. On that same day the admission fees collected totaled $792. How many children were admitted on that day? How many adults were admitted?
Equations
$x + y = 278$ $1.5x + 4y = 792$
Solution
$x = 128 \quad y = 150$

Figure 1: An example algebra word problem. Our goal is to map a given problem to a set of equations representing its algebraic meaning, which are then solved to get the problem's answer.

Our model defines a joint log-linear distribution over full systems of equations and alignments between these equations and the text. The space of possible equations is defined by a set of equation templates, which we induce from the training examples, where each template has a set of slots. Number slots are filled by numbers from the text, and unknown slots are aligned to nouns. For example, the system in Figure 1 is generated by filling one such template with four specific numbers (1.5, 4, 278, and 792) and aligning two nouns ("Tickets" in "Tickets for children", and "tickets" in "Adult tickets"). These inferred correspondences are used to define cross-sentence features that provide global cues to the model. For instance, in our running example, the string

[1] The code and data for this work are available at http://groups.csail.mit.edu/rbg/code/wordprobs/.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 271–281,
Baltimore, Maryland, USA, June 23-25 2014.

pairs ("$1.50", "children") and ("$4","adults") both surround the word "cost," suggesting an output equation with a sum of two constant-variable products.

We consider learning with two different levels of supervision. In the first scenario, we assume access to each problem's numeric solution (see Figure 1) for most of the data, along with a small set of seed examples labeled with full equations. During learning, a solver evaluates competing hypotheses to drive the learning process. In the second scenario, we are provided with a full system of equations for each problem. In both cases, the available labeled equations (either the seed set, or the full set) are abstracted to provide the model's equation templates, while the slot filling and alignment decisions are latent variables whose settings are estimated by directly optimizing the marginal data log-likelihood.

The approach is evaluated on a new corpus of 514 algebra word problems and associated equation systems gathered from Algebra.com. Provided with full equations during training, our algorithm successfully solves over 69% of the word problems from our test set. Furthermore, we find the algorithm can robustly handle weak supervision, achieving more than 70% of the above performance when trained exclusively on answers.

2 Related Work

Our work is related to three main areas of research: situated semantic interpretation, information extraction, and automatic word problem solvers.

Situated Semantic Interpretation There is a large body of research on learning to map natural language to formal meaning representations, given varied forms of supervision. Reinforcement learning can be used to learn to read instructions and perform actions in an external world (Branavan et al., 2009; Branavan et al., 2010; Vogel and Jurafsky, 2010). Other approaches have relied on access to more costly annotated logical forms (Zelle and Mooney, 1996; Thompson and Mooney, 2003; Wong and Mooney, 2006; Zettlemoyer and Collins, 2005; Kwiatkowski et al., 2010). These techniques have been generalized more recently to learn from sentences paired with indirect feedback from a controlled application. Examples include question answering (Clarke et al., 2010; Cai and Yates, 2013a; Cai and Yates, 2013b; Berant et al., 2013; Kwiatkowski et al., 2013), dialog systems (Artzi and Zettlemoyer, 2011), robot instruction (Chen and Mooney, 2011; Chen, 2012; Kim and Mooney, 2012; Matuszek et al., 2012; Artzi and Zettlemoyer, 2013), and program executions (Kushman and Barzilay, 2013; Lei et al., 2013). We focus on learning from varied supervision, including question answers and equation systems, both can be obtained reliably from annotators with no linguistic training and only basic math knowledge.

Nearly all of the above work processed single sentences in isolation. Techniques that consider multiple sentences typically do so in a serial fashion, processing each in turn with limited cross-sentence reasoning (Branavan et al., 2009; Zettlemoyer and Collins, 2009; Chen and Mooney, 2011; Artzi and Zettlemoyer, 2013). We focus on analyzing multiple sentences simultaneously, as is necessary to generate the global semantic representations common in domains such as algebra word problems.

Information Extraction Our approach is related to work on template-based information extraction, where the goal is to identify instances of event templates in text and extract their slot fillers. Most work has focused on the supervised case, where the templates are manually defined and data is labeled with alignment information, e.g. (Grishman et al., 2005; Maslennikov and Chua, 2007; Ji and Grishman, 2008; Reichart and Barzilay, 2012). However, some recent work has studied the automatic induction of the set of possible templates from data (Chambers and Jurafsky, 2011; Ritter et al., 2012). In our approach, systems of equations are relatively easy to specify, providing a type of template structure, and the alignment of the slots in these templates to the text is modeled primarily with latent variables during learning. Additionally, mapping to a semantic representation that can be executed allows us to leverage weaker supervision during learning.

Automatic Word Problem Solvers Finally, there has been research on automatically solving various types of mathematical word problems. The dominant existing approach is to hand engineer rule-based systems to solve math problem in specific domains (Mukherjee and Garain, 2008; Lev et al., 2004). Our focus is on learning a model for the end-to-end task of solving word problems given only a training corpus of questions paired with equations or answers.

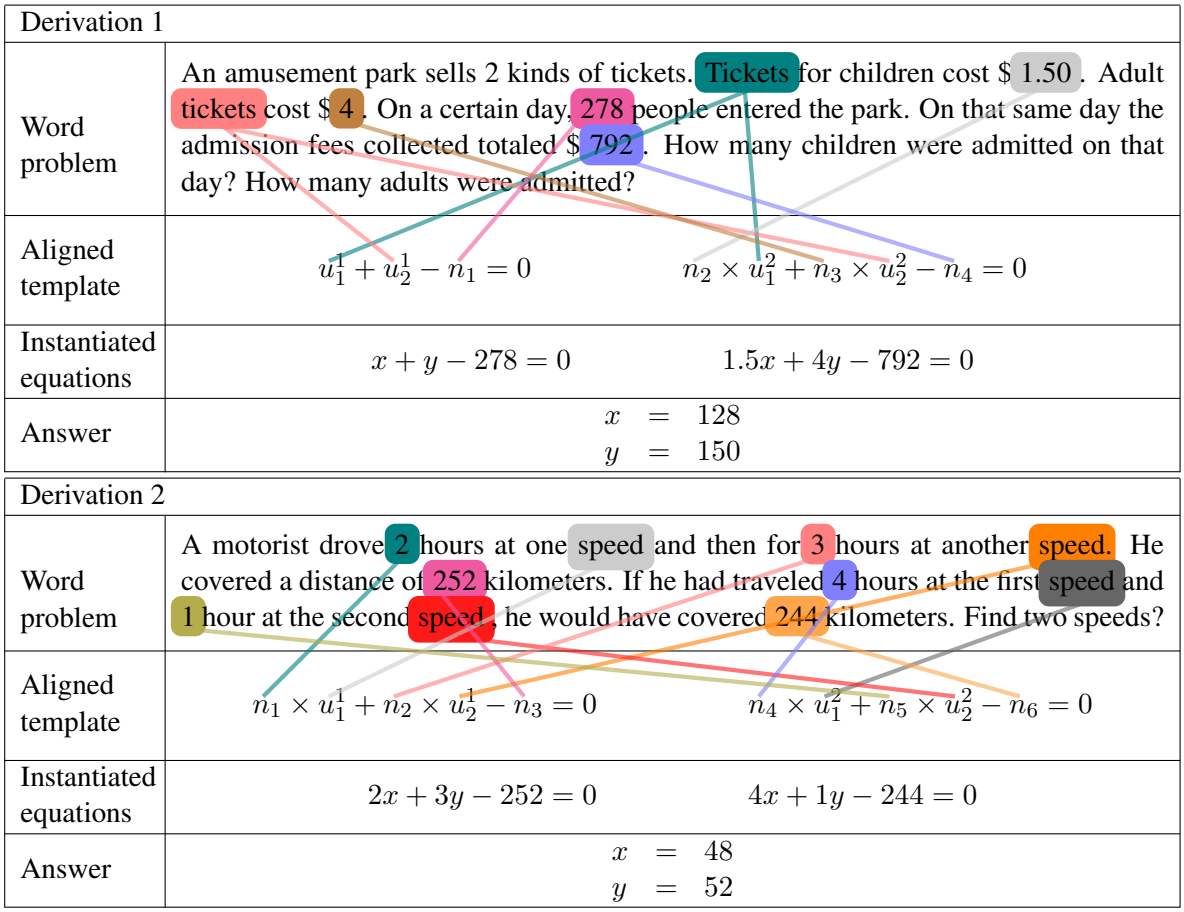


Figure 2: Two complete derivations for two different word problems. Derivation 1 shows an alignment where two instances of the same slot are aligned to the same word (e.g., u_1^1 and u_1^2 both are aligned to "Tickets"). Derivation 2 includes an alignment where four identical nouns are each aligned to different slot instances in the template (e.g., the first "speed" in the problem is aligned to u_1^1).

3 Mapping Word Problems to Equations

We define a two step process to map word problems to equations. First, a template is selected to define the overall structure of the equation system. Next, the template is instantiated with numbers and nouns from the text. During inference we consider these two steps jointly.

Figure 2 shows both steps for two derivations. The template dictates the form of the equations in the system and the type of slots in each: u slots represent unknowns and n slots are for numbers that must be filled from the text. In Derivation 1, the selected template has two unknown slots, u_1 and u_2, and four number slots, n_1 to n_4. Slots can be shared between equations, for example, the unknown slots u_1 and u_2 in the example appear in both equations. A slot may have different instances, for example u_1^1 and u_1^2 are the two instances of u_1 in the example.

We align each slot instance to a word in the problem. Each number slot n is aligned to a number, and each unknown slot u is aligned to a noun. For example, Derivation 1 aligns the number 278 to n_1, 1.50 to n_2, 4 to n_3, and 792 to n_4. It also aligns both instances of u_1 (e.g., u_1^1 and u_1^2) to "Tickets", and both instances of u_2 to "tickets". In contrast, in Derivation 2, instances of the same unknown slot (e.g. u_1^1 and u_1^2) are aligned to two different words in the problem (different occurrences of the word "speed"). This allows for a tighter mapping between the natural language and the system template, where the words aligned to the first equation in the template come from the first two sentences, and the words aligned to the second equation come from the third.

Given an alignment, the template can then be instantiated: each number slot n is replaced with the aligned number, and each unknown slot u with a variable. This output system of equations is then automatically solved to generate the final answer.

3.1 Derivations

Definitions Let $\mathcal{X}$ be the set of all word problems. A word problem $x \in \mathcal{X}$ is a sequence of k words $\langle w_1, \ldots w_k \rangle$. Also, define an *equation template* t to be a formula $A = B$, where A and B are expressions. An expression A is one of the following:

- A number constant f.
- A number slot n.
- An unknown slot u.
- An application of a mathematical relation R to two expressions (e.g., $n_1 \times u_1$).

We define a *system template* T to be a set of l equation templates $\{t_0, \ldots, t_l\}$. $\mathcal{T}$ is the set of all system templates. A slot may occur more than once in a system template, to allow variables to be reused in different equations. We denote a specific instance i of a slot, u for example, as u^i. For brevity, we omit the instance index when a slot appears only once. To capture a correspondence between the text of x and a template T, we define an alignment p to be a set of pairs (w, s), where w is a token in x and s is a slot instance in T.

Given the above definitions, an equation e can be constructed from a template t where each number slot n is replaced with a real number, each unknown slot u is replaced with a variable, and each number constant f is kept as is. We call the process of turning a template into an equation *template instantiation*. Similarly, an equation system E is a set of l equations $\{e_0, \ldots, e_l\}$, which can be constructed by instantiating each of the equation templates in a system template T. Finally, an answer a is a tuple of real numbers.

We define a derivation y from a word problem to an answer as a tuple (T, p, a), where T is the selected system template, p is an alignment between T and x, and a is the answer generated by instantiating T using x through p and solving the generated equations. Let $\mathcal{Y}$ be the set of all derivations.

The Space of Possible Derivations We aim to map each word problem x to an equation system E. The space of equation systems considered is defined by the set of possible system templates $\mathcal{T}$ and the words in the original problem x, that are available for filling slots. In practice, we generate $\mathcal{T}$ from the training data, as described in Section 4.1. Given a system template $T \in \mathcal{T}$, we create an alignment p between T and x. The set of possible alignment pairs is constrained as follows: each number slot $n \in T$ can be aligned to any number in the text, a number word can only be aligned to a single slot n, and must be aligned to all instances of that slot. Additionally, an unknown slot instance $u \in T$ can only be aligned to a noun word. A complete derivation's alignment pairs all slots in T with words in x.

An **amusement park** sells 2 **kinds** of **tickets**. **Tickets** for **children** cost **$** 1.50. Adult **tickets** cost **$** 4. On a certain **day**, 278 **people** entered the **park**. On that same **day** the **admission fees** collected totaled **$** 792. How many **children** were admitted on that **day**? How many **adults** were admitted?

$$\begin{aligned} u_1^1 + u_2^1 - n_1 &= 0 \\ n_2 \times u_1^2 + n_3 \times u_2^2 - n_4 &= 0 \end{aligned}$$

Figure 3: The first example problem and selected system template from Figure 2 with all potential aligned words marked. Nouns (boldfaced) may be aligned to unknown slot instances u_i^j, and number words (highlighted) may be aligned to number slots n_i.

Figure 3 illustrates the space of possible alignments for the first problem and system template from Figure 2. Nouns (shown in boldface) can be aligned to any of the unknown slot instances in the selected template (u_1^1, u_1^2, u_2^1, and u_2^2 for the template selected). Numbers (highlighted) can be aligned to any of the number slots (n_1, n_2, n_3, and n_4 in the template).

3.2 Probabilistic Model

Due to the ambiguity in selecting the system template and alignment, there will be many possible derivations $y \in \mathcal{Y}$ for each word problem $x \in \mathcal{X}$. We discriminate between competing analyses using a log-linear model, which has a feature function $\phi : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^d$ and a parameter vector $\theta \in \mathbb{R}^d$. The probability of a derivation y given a problem x is defined as:

$$p(y|x;\theta) = \frac{e^{\theta \cdot \phi(x,y)}}{\sum_{y' \in \mathcal{Y}} e^{\theta \cdot \phi(x,y')}}$$

Section 6 defines the full set of features used.

The inference problem at test time requires us to find the most likely answer a given a problem

x, assuming the parameters θ are known:

$$f(x) = \arg\max_a p(a|x;\theta)$$

Here, the probability of the answer is marginalized over template selection and alignment:

$$p(a|x;\theta) = \sum_{\substack{y \in \mathcal{Y} \\ \text{s.t. } \text{AN}(y)=a}} p(y|x;\theta) \quad (1)$$

where $\text{AN}(y)$ extracts the answer a out of derivation y. In this way, the distribution over derivations y is modeled as a latent variable. We use a beam search inference procedure to approximately compute Equation 1, as described in Section 5.

4 Learning

To learn our model, we need to induce the structure of system templates in $\mathcal{T}$ and estimate the model parameters θ.

4.1 Template Induction

It is possible to generate system templates $\mathcal{T}$ when provided access to a set of n training examples $\{(x_i, E_i) : i = 1, \ldots, n\}$, where x_i is a word problem and E_i is a set of equations. We generalize each E to a system template T by (a) replacing each variable with an unknown slot, and (b) replacing each number mentioned in the text with a number slot. Numbers not mentioned in the problem text remain in the template as constants. This allows us to solve problems that require numbers that are implied by the problem semantics rather than appearing directly in the text, such as the percent problem in Figure 4.

4.2 Parameter Estimation

For parameter estimation, we assume access to n training examples $\{(x_i, \mathcal{V}_i) : i = 1, \ldots, n\}$, each containing a word problem x_i and a validation function $\mathcal{V}_i$. The validation function $\mathcal{V} : \mathcal{Y} \rightarrow \{0, 1\}$ maps a derivation $y \in \mathcal{Y}$ to 1 if it is correct, or 0 otherwise.

We can vary the validation function to learn from different types of supervision. In Section 8, we will use validation functions that check whether the derivation y has either (1) the correct system of equations E, or (2) the correct answer a. Also, using different types of validation functions on different subsets of the data enables semi-supervised learning. This approach is related to Artzi and Zettlemoyer (2013).

Word problem
A chemist has a solution that is 18 % alcohol and one that is 50 % alcohol. He wants to make 80 liters of a 30 % solution. How many liters of the 18 % solution should he add? How many liters of the 30 % solution should he add?
Labeled equations
$18 \times 0.01 \times x + 50 \times 0.01 \times y = 30 \times 0.01 \times 80$ $x + y = 80$
Induced template system
$n_1 \times 0.01 \times u_1^1 + n_2 \times 0.01 \times u_2^1 = n_3 \times 0.01 \times n_4$ $u_1^2 + u_2^2 = n_5$

Figure 4: During template induction, we automatically detect the numbers in the problem (highlighted above) to generalize the labeled equations to templates. Numbers not present in the text are considered part of the induced template.

We estimate θ by maximizing the conditional log-likelihood of the data, marginalizing over all valid derivations:

$$O = \sum_i \sum_{\substack{y \in \mathcal{Y} \\ \text{s.t. } \mathcal{V}_i(y)=1}} \log p(y|x_i;\theta)$$

We use L-BFGS (Nocedal and Wright, 2006) to optimize the parameters. The gradient of the individual parameter θ_j is given by:

$$\frac{\partial O}{\partial \theta_j} = \sum_i E_{p(y|x_i,\mathcal{V}_i(y)=1;\theta)}\left[\phi_j(x_i, y)\right] - E_{p(y|x_i;\theta)}\left[\phi_j(x_i, y)\right] \quad (2)$$

Section 5 describes how we approximate the two terms of the gradient using beam search.

5 Inference

Computing the normalization constant for Equation 1 requires summing over all templates and all possible ways to instantiate them. This results in a search space exponential in the number of slots in the largest template in $\mathcal{T}$, the set of available system templates. Therefore, we approximate this computation using beam search. We initialize the beam with all templates in $\mathcal{T}$ and iteratively align slots from the templates in the beam to words in the problem text. For each template, the next slot

to be considered is selected according to a predefined canonicalized ordering for that template. After each iteration we prune the beam to keep the top-k partial derivations according to the model score. When pruning the beam, we allow at most l partial derivations for each template, to ensure that a small number of templates don't monopolize the beam. We continue this process until all templates in the beam are fully instantiated.

During learning we compute the second term in the gradient (Equation 2) using our beam search approximation. Depending on the available validation function $\mathcal{V}$ (as defined in Section 4.2), we can also accurately prune the beam for the computation of the first half of the gradient. Specifically, when assuming access to labeled equations, we can constrain the search to consider only partial hypotheses that could possibly be completed to produce the labeled equations.

6 Model Details

Template Canonicalization There are many syntactically different but semantically equivalent ways to express a given system of equations. For example, the phrase "John is 3 years older than Bill" can be written as $j = b + 3$ or $j - 3 = b$. To avoid such ambiguity, we canonicalize templates into a normal form representation. We perform this canonicalization by obtaining the symbolic solution for the unknown slots in terms of the number slots and constants using the mathematical solver Maxima (Maxima, 2014).

Slot Signature In a template like $s_1 + s_2 = s_3$, the slot s_1 is distinct from the slot s_2, but we would like them to share many of the features used in deciding their alignment. To facilitate this, we generate signatures for each slot and slot pair. The signature for a slot indicates the system of equations it appears in, the specific equation it is in, and the terms of the equation it is a part of. Pairwise slot signatures concatenate the signatures for the two slots as well as indicating which terms are shared. This allows, for example, n_2 and n_3 in Derivation 1 in Figure 2 to have the same signature, while the pairs $\langle n_2, u_1 \rangle$ and $\langle n_3, u_1 \rangle$ have different ones. To share features across templates, slot and slot-pair signatures are generated for both the full template, as well as for each of the constituent equations.

Features The features $\phi(x, y)$ are computed for a derivation y and problem x and cover all derivation decisions, including template and alignment selection. When required, we use standard tools to generate part-of-speech tags, lematizations, and dependency parses to compute features.[2] For each number word in y we also identify the closest noun in the dependency parse. For example, the noun for 278 in Derivation 1, Figure 2 would be "people." The features are calculated based on these nouns, rather than the number words.

Document level
Unigrams
Bigrams
Single slot
Has the same lemma as a question object
Is a question object
Is in a question sentence
Is equal to one or two (for numbers)
Word lemma X nearby constant
Slot pair
Dep. path contains: Word
Dep. path contains: Dep. Type
Dep. path contains: Word X Dep. Type
Are the same word instance
Have the same lemma
In the same sentence
In the same phrase
Connected by a preposition
Numbers are equal
One number is larger than the other
Equivalent relationship
Solution Features
Is solution all positive
Is solution all integer

Table 1: The features divided into categories.

We use four types of features: document level features, features that look at a single slot entry, features that look at pairs of slot entries, and features that look at the numeric solutions. Table 1 lists all the features used. Unless otherwise noted, when computing slot and slot pair features, a separate feature is generated for each of the signature types discussed earlier.

Document level features Oftentimes the natural language in x will contain words or phrases which are indicative of a certain template, but are not associated with any of the words aligned to slots in the template. For example, the word "chemist"

[2] In our experiments these are generated using the Stanford parser (de Marneffe et al., 2006)

might indicate a template like the one seen in Figure 4. We include features that connect each template with the unigrams and bigrams in the word problem. We also include an indicator feature for each system template, providing a bias for its use.

Single Slot Features The natural language x always contains one or more questions or commands indicating the queried quantities. For example, the first problem in Figure 2 asks "How many children were admitted on that day?" The queried quantities, the number of children in this case, must be represented by an unknown in the system of equations. We generate a set of features which look at both the word overlap and the noun phrase overlap between slot words and the objects of a question or command sentence. We also compute a feature indicating whether a slot is filled from a word in a question sentence. Additionally, algebra problems frequently use phrases such as "2 kinds of tickets" (e.g., Figure 2). These numbers do not typically appear in the equations. To account for this, we add a single feature indicating whether a number is one or two. Lastly, many templates contain constants which are identifiable from words used in nearby slots. For example, in Figure 4 the constant 0.01 is related to the use of "%" in the text. To capture such usage, we include a set of lexicalized features which concatenate the word lemma with nearby constants in the equation. These features do not include the slot signature.

Slot Pair Features The majority of features we compute account for relationships between slot words. This includes features that trigger for various equivalence relations between the words themselves, as well as features of the dependency path between them. We also include features that look at the numerical relationship of two numbers, where the numeric values of the unknowns are generated by solving the system of equations. This helps recognize that, for example, the total of a sum is typically larger than each of the (typically positive) summands.

Additionally, we also have a single feature looking at shared relationships between pairs of slots. For example, in Figure 2 the relationship between "tickets for children" and "$1.50" is "cost". Similarly the relationship between "Adult tickets" and "$4" is also "cost". Since the actual nature of this relationship is not important, this feature is not lexicalized, instead it is only triggered for the presence of equality. We consider two cases: subject-object relationships where the intervening verb is equal, and noun-to-preposition object relationships where the intervening preposition is equal.

# problems	514
# sentences	1616
# words	19357
Vocabulary size	2352
Mean words per problem	37
Mean sentences per problem	3.1
Mean nouns per problem	13.4
# unique equation systems	28
Mean slots per system	7
Mean derivations per problem	4M

Table 2: Dataset statistics.

Solution Features By grounding our semantics in math, we are able to include features which look at the final answer, a, to learn which answers are reasonable for the algebra problems we typically see. For example, the solution to many, but not all, of the problems involves the size of some set of objects which must be both positive and integer.

7 Experimental Setup

Dataset We collected a new dataset of algebra word problems from Algebra.com, a crowdsourced tutoring website. The questions were posted by students for members of the community to respond with solutions. Therefore, the problems are highly varied, and are taken from real problems given to students. We heuristically filtered the data to get only linear algebra questions which did not require any explicit background knowledge. From these we randomly chose a set of 1024 questions. As the questions are posted to a web forum, the posts often contained additional comments which were not part of the word problems and the solutions are embedded in long free-form natural language descriptions. To clean the data we asked Amazon Mechanical Turk workers to extract from the text: the algebra word problem itself, the solution equations, and the numeric answer. We manually verified both the equations and the numbers to ensure they were correct. To ensure each problem type is seen at least a few times in the training data, we removed the infrequent problem types. Specifically, we induced the system template from each equation system, as described in Section 4.1, and removed all problems for which the associated system template appeared

less than 6 times in the dataset. This left us with 514 problems. Table 2 provides the data statistics.

Forms of Supervision We consider both semi-supervised and supervised learning. In the semi-supervised scenario, we assume access to the numerical answers of all problems in the training corpus and to a small number of problems paired with full equation systems. To select which problems to annotate with equations, we identified the five most common types of questions in the data and annotated a randomly sampled question of each type. 5EQ+ANS uses this form of weak supervision. To show the benefit of using the weakly supervised data, we also provide results for a baseline scenario 5EQ, where the training data includes *only* the five seed questions annotated with equation systems. In the fully supervised scenario ALLEQ, we assume access to full equation systems for the entire training set.

Evaluation Protocol We run all our experiments using 5-fold cross-validation. Since our model generates a solution for every problem, we report only accuracy. We report two metrics: equation accuracy to measure how often the system generates the correct equation system, and answer accuracy to evaluate how often the generated numerical answer is correct. When comparing equations, we avoid spurious differences by canonicalizing the equation system, as described in Section 6. To compare answer tuples we disregard the ordering and require each number appearing in the reference answer to appear in the generated answer.

Parameters and Solver In our experiments we set k in our beam search algorithm (Section 5) to 200, and l to 20. We run the L-BFGS computation for 50 iterations. We regularize our learning objective using the L^2-norm and a λ value of 0.1. The set of mathematical relations supported by our implementation is $\{+, -, \times, /\}$.Our implementation uses the Gaussian Elimination function in the Efficient Java Matrix Library (EJML) (Abeles, 2014) to generate answers given a set of equations.

8 Results

8.1 Impact of Supervision

Table 3 summarizes the results. As expected, having access to the full system of equations (ALLEQ) at training time results in the best learned model, with nearly 69% accuracy. However, training from primarily answer annotations (5EQ+ANS) results in performance which is almost 70% of ALLEQ, demonstrating the value of weakly supervised data. In contrast, 5EQ, which cannot use this weak supervision, performs much worse.

	Equation accuracy	Answer accuracy
5EQ	20.4	20.8
5EQ+ANS	45.7	46.1
ALLEQ	66.1	68.7

Table 3: Cross-validation accuracy results for various forms of supervision.

	Equation accuracy	Answer accuracy	% of data
≤ 10	43.6	50.8	25.5
$11 - 15$	46.6	45.1	10.5
$16 - 20$	44.2	52.0	11.3
> 20	85.7	86.1	52.7

Table 4: Performance on different template frequencies for ALLEQ.

8.2 Performance and Template Frequency

To better understand the results, we also measured equation accuracy as a function of the frequency of each equation template in the data set. Table 4 reports results for ALLEQ after grouping the problems into four different frequency bins. We can see that the system correctly answers more than 85% of the question types which occur frequently while still achieving more than 50% accuracy on those that occur relatively infrequently. We do not include template frequency results for 5EQ+ANS since in this setup our system is given only the top five most common templates. This limited set of templates covers only those questions in the > 20 bin, or about 52% of the data. However, on this subset 5EQ+ANS performs very well, answering 88% of them correctly, which is approximately the same as the 86% achieved by ALLEQ. Thus while the weak supervision is not helpful in generating the space of possible equations, it is very helpful in learning to generate the correct answer when given an appropriate space of equations.

8.3 Ablation Analysis

Table 5 shows ablation results for each group of features. The results along the diagonal show the performance when a single group of features is ablated, while the off-diagonal numbers show the

	w/o pair	w/o document	w/o solution	w/o single
w/o pair	42.8	25.7	19.0	39.6
w/o document	–	63.8	50.4	57.6
w/o solution	–	–	63.6	62.0
w/o single	–	–	–	65.9

Table 5: Cross-validation accuracy results with different feature groups ablated for ALLEQ. Results are for answer accuracy which is 68.7% without any features ablated.

performance when two groups of features are ablated together. We can see that all of the features contribute to the overall performance, and that the pair features are the most important followed by the document and solution features. We also see that the pair features can compensate for the absence of other features. For example, the performance drops only slightly when either the document or solution features are removed in isolation. However, the drop is much more dramatic when they are removed along with the pair features.

8.4 Qualitative Error Analysis

We examined our system output on one fold of ALLEQ and identified two main classes of errors.

The first, accounting for approximately one-quarter of the cases, includes mistakes where more background or world knowledge might have helped. For example, Problem 1 in Figure 5 requires understanding the relation between the dimensions of a painting, and how this relation is maintained when the painting is printed, and Problem 2 relies on understanding concepts of commerce, including cost, sale price, and profit. While these relationships could be learned in our model with enough data, as it does for percentage problems (e.g., Figure 4), various outside resources, such as knowledge bases (e.g. Freebase) or distributional statistics from a large text corpus, might help us learn them with less training data.

The second category, which accounts for about half of the errors, includes mistakes that stem from compositional language. For example, the second sentence in Problem 3 in Figure 5 could generate the equation $2x - y = 5$, with the phrase "twice of one of them" generating the expression $2x$. Given the typical shallow nesting, it's possible to learn templates for these cases given enough data, and in the future it might also be possible to develop new, cross-sentence semantic parsers to enable better generalization from smaller datasets.

(1)	A painting is 10 inches tall and 15 inches wide. A print of the painting is 25 inches tall, how wide is the print in inches?
(2)	A textbook costs a bookstore 44 dollars, and the store sells it for 55 dollars. Find the amount of profit based on the selling price.
(3)	The sum of two numbers is 85. The difference of twice of one of them and the other one is 5. Find both numbers.
(4)	The difference between two numbers is 6. If you double both numbers, the sum is 36. Find the two numbers.

Figure 5: Examples of problems our system does not solve correctly.

9 Conclusion

We presented an approach for automatically learning to solve algebra word problems. Our algorithm constructs systems of equations, while aligning their variables and numbers to the problem text. Using a newly gathered corpus we measured the effects of various forms of weak supervision on performance. To the best of our knowledge, we present the first learning result for this task.

There are still many opportunities to improve the reported results, and extend the approach to related domains. We would like to develop techniques to learn compositional models of meaning for generating new equations. Furthermore, the general representation of mathematics lends itself to many different domains including geometry, physics, and chemistry. Eventually, we hope to extend the techniques to synthesize even more complex structures, such as computer programs, from natural language.

Acknowledgments

The authors acknowledge the support of Battelle Memorial Institute (PO#300662) and NSF (grant IIS-0835652). We thank Nicholas FitzGerald, the MIT NLP group, the UW NLP group and the ACL reviewers for their suggestions and comments. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors, and do not necessarily reflect the views of the funding organizations.

References

Peter Abeles. 2014. Efficient java matrix library. https://code.google.com/p/efficient-java-matrix-library/.

Yoav Artzi and Luke Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

S.R.K Branavan, Harr Chen, Luke Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics.*

S.R.K Branavan, Luke Zettlemoyer, and Regina Barzilay. 2010. Reading between the lines: Learning to map high-level instructions to commands. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics.*

Qingqing Cai and Alexander Yates. 2013a. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Qingqing Cai and Alexander Yates. 2013b. Semantic parsing freebase: Towards open-domain semantic parsing. In *Proceedings of the Joint Conference on Lexical and Computational Semantics.*

Nathanael Chambers and Dan Jurafsky. 2011. Template-based information extraction without the templates. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

David Chen and Raymond Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the Conference on Artificial Intelligence.*

David Chen. 2012. Fast online lexicon learning for grounded language acquisition. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of the Conference on Computational Natural Language Learning.* Association for Computational Linguistics.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Conference on Language Resources and Evaluation*.

Ralph Grishman, David Westbrook, and Adam Meyers. 2005. NYUs English ACE 2005 System Description. In *Proceedings of the Automatic Content Extraction Evaluation Workshop*.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Joohyun Kim and Raymond Mooney. 2012. Unsupervised pcfg induction for grounded language learning with highly ambiguous supervision. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Nate Kushman and Regina Barzilay. 2013. Using semantic unification to generate regular expressions from natural language. In *Proceeding of the Annual Meeting of the North American Chapter of the Association for Computational Linguistics*.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic ccg grammars from logical form with higher-order unification. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing.*

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of Empirical Methods in Natural Language Processing*.

Tao Lei, Fan Long, Regina Barzilay, and Martin Rinard. 2013. From natural language specifications to program input parsers. In *Proceeding of the Association for Computational Linguistics.*

Iddo Lev, Bill MacCartney, Christopher Manning, and Roger Levy. 2004. Solving logic puzzles: From robust processing to precise semantics. In *Proceedings of the Workshop on Text Meaning and Interpretation.* Association for Computational Linguistics.

Mstislav Maslennikov and Tat-Seng Chua. 2007. A multi-resolution framework for information extraction from free text. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Cynthia Matuszek, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. 2012. A joint model of language and perception for grounded attribute learning. In *Proceedings of the International Conference on Machine Learning*.

Maxima. 2014. Maxima, a computer algebra system. version 5.32.1.

Anirban Mukherjee and Utpal Garain. 2008. A review of methods for automatic understanding of natural language mathematical problems. *Artificial Intelligence Review*, 29(2).

Jorge Nocedal and Stephen Wright. 2006. Numerical optimization, series in operations research and financial engineering. *Springer, New York*.

Roi Reichart and Regina Barzilay. 2012. Multi-event extraction guided by global constraints. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*.

Alan Ritter, Mausam, Oren Etzioni, and Sam Clark. 2012. Open domain event extraction from twitter. In *Proceedings of the Conference on Knowledge Discovery and Data Mining*.

Cynthia Thompson and Raymond Mooney. 2003. Acquiring word-meaning mappings for natural language interfaces. *Journal of Artificial Intelligence Research*, 18(1).

Adam Vogel and Dan Jurafsky. 2010. Learning to follow navigational directions. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Yuk Wah Wong and Raymond Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Annual Meeting of the North American Chapter of the Association of Computational Linguistics*. Association for Computational Linguistics.

John Zelle and Raymond Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the Conference on Artificial Intelligence*.

Luke Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

Luke Zettlemoyer and Michael Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proceedings of the Joint Conference of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing*.

Modelling function words improves unsupervised word segmentation

Mark Johnson[1,2], **Anne Christophe**[3,4], **Katherine Demuth**[2,6] and **Emmanuel Dupoux**[3,5]

[1] Department of Computing, Macquarie University, Sydney, Australia

[2] Santa Fe Institute, Santa Fe, New Mexico, USA

[3] Ecole Normale Supérieure, Paris, France

[4] Centre National de la Recherche Scientifique, Paris, France

[5] Ecole des Hautes Etudes en Sciences Sociales, Paris, France

[6] Department of Linguistics, Macquarie University, Sydney, Australia

Abstract

Inspired by experimental psychological findings suggesting that function words play a special role in word learning, we make a simple modification to an Adaptor Grammar based Bayesian word segmentation model to allow it to learn sequences of monosyllabic "function words" at the beginnings and endings of collocations of (possibly multi-syllabic) words. This modification improves unsupervised word segmentation on the standard Bernstein-Ratner (1987) corpus of child-directed English by more than 4% token f-score compared to a model identical except that it does not special-case "function words", setting a new state-of-the-art of 92.4% token f-score. Our function word model assumes that function words appear at the left periphery, and while this is true of languages such as English, it is not true universally. We show that a learner can use Bayesian model selection to determine the location of function words in their language, even though the input to the model only consists of unsegmented sequences of phones. Thus our computational models support the hypothesis that function words play a special role in word learning.

1 Introduction

Over the past two decades psychologists have investigated the role that function words might play in human language acquisition. Their experiments suggest that function words play a special role in the acquisition process: children learn function words before they learn the vast bulk of the associated content words, and they use function words to help identify context words.

The goal of this paper is to determine whether computational models of human language acquisition can provide support for the hypothesis that function words are treated specially in human language acquisition. We do this by comparing two computational models of word segmentation which differ solely in the way that they model function words. Following Elman et al. (1996) and Brent (1999) our word segmentation models identify word boundaries from unsegmented sequences of phonemes corresponding to utterances, effectively performing unsupervised learning of a lexicon. For example, given input consisting of unsegmented utterances such as the following:

j u w a n t t u s i ð ə b ʊ k

a word segmentation model should segment this as *ju want tu si ðə bʊk*, which is the IPA representation of "you want to see the book".

We show that a model equipped with the ability to learn some rudimentary properties of the target language's function words is able to learn the vocabulary of that language more accurately than a model that is identical except that it is incapable of learning these generalisations about function words. This suggests that there are acquisition advantages to treating function words specially that human learners could take advantage of (at least to the extent that they are learning similar generalisations as our models), and thus supports the hypothesis that function words are treated specially in human lexical acquisition. As a reviewer points out, we present no evidence that children use function words in the way that our model does, and we want to emphasise we make no such claim. While absolute accuracy is not directly relevant to the main point of the paper, we note that the models that learn generalisations about function words perform unsupervised word segmentation at 92.5% token f-score on the standard Bernstein-Ratner (1987) corpus, which improves the previous state-of-the-art by more than 4%.

As a reviewer points out, the changes we make to our models to incorporate function words can be viewed as "building in" substantive information about possible human languages. The model

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 282–292,
Baltimore, Maryland, USA, June 23-25 2014.

that achieves the best token f-score expects function words to appear at the left edge of phrases. While this is true for languages such as English, it is not true universally. By comparing the posterior probability of two models — one in which function words appear at the left edges of phrases, and another in which function words appear at the right edges of phrases — we show that a learner could use Bayesian posterior probabilities to determine that function words appear at the left edges of phrases in English, even though they are not told the locations of word boundaries or which words are function words.

This paper is structured as follows. Section 2 describes the specific word segmentation models studied in this paper, and the way we extended them to capture certain properties of function words. The word segmentation experiments are presented in section 3, and section 4 discusses how a learner could determine whether function words occur on the left-periphery or the right-periphery in the language they are learning. Section 5 concludes and describes possible future work. The rest of this introduction provides background on function words, the Adaptor Grammar models we use to describe lexical acquisition and the Bayesian inference procedures we use to infer these models.

1.1 Psychological evidence for the role of function words in word learning

Traditional descriptive linguistics distinguishes *function words*, such as determiners and prepositions, from *content words*, such as nouns and verbs, corresponding roughly to the distinction between functional categories and lexical categories of modern generative linguistics (Fromkin, 2001).

Function words differ from content words in at least the following ways:

1. there are usually far fewer function word types than content word types in a language
2. function word types typically have much higher token frequency than content word types
3. function words are typically morphologically and phonologically simple (e.g., they are typically monosyllabic)
4. function words typically appear in peripheral positions of phrases (e.g., prepositions typically appear at the beginning of prepositional phrases)
5. each function word class is associated with specific content word classes (e.g., determiners and prepositions are associated with nouns, auxiliary verbs and complementisers are associated with main verbs)
6. semantically, content words denote sets of objects or events, while function words denote more complex relationships over the entities denoted by content words
7. historically, the rate of innovation of function words is much lower than the rate of innovation of content words (i.e., function words are typically "closed class", while content words are "open class")

Properties 1–4 suggest that function words might play a special role in language acquisition because they are especially easy to identify, while property 5 suggests that they might be useful for identifying lexical categories. The models we study here focus on properties 3 and 4, in that they are capable of learning specific sequences of monosyllabic words in peripheral (i.e., initial or final) positions of phrase-like units.

A number of psychological experiments have shown that infants are sensitive to the function words of their language within their first year of life (Shi et al., 2006; Hallé et al., 2008; Shafer et al., 1998), often before they have experienced the "word learning spurt". Crucially for our purpose, infants of this age were shown to exploit frequent function words to segment neighboring content words (Shi and Lepage, 2008; Hallé et al., 2008). In addition, 14 to 18-month-old children were shown to exploit function words to constrain lexical access to known words - for instance, they expect a noun after a determiner (Cauvet et al., 2014; Kedar et al., 2006; Zangl and Fernald, 2007). In addition, it is plausible that function words play a crucial role in children's acquisition of more complex syntactic phenomena (Christophe et al., 2008; Demuth and McCullough, 2009), so it is interesting to investigate the roles they might play in computational models of language acquisition.

1.2 Adaptor grammars

Adaptor grammars are a framework for Bayesian inference of a certain class of hierarchical non-parametric models (Johnson et al., 2007b). They define distributions over the trees specified by a context-free grammar, but unlike probabilistic context-free grammars, they "learn" distributions over the possible subtrees of a user-specified set of "adapted" nonterminals. (Adaptor grammars are non-parametric, i.e., not characterisable by a finite

set of parameters, if the set of possible subtrees of the adapted nonterminals is infinite). Adaptor grammars are useful when the goal is to learn a potentially unbounded set of entities that need to satisfy hierarchical constraints. As section 2 explains in more detail, word segmentation is such a case: words are composed of syllables and belong to phrases or collocations, and modelling this structure improves word segmentation accuracy.

Adaptor Grammars are formally defined in Johnson et al. (2007b), which should be consulted for technical details. Adaptor Grammars (AGs) are an extension of Probabilistic Context-Free Grammars (PCFGs), which we describe first. A *Context-Free Grammar* (CFG) $G = (N, W, R, S)$ consists of disjoint finite sets of *nonterminal symbols* N and *terminal symbols* W, a finite set of *rules* R of the form $A \rightarrow \alpha$ where $A \in N$ and $\alpha \in (N \cup W)^\star$, and a *start symbol* $S \in N$. (We assume there are no "ϵ-rules" in R, i.e., we require that $|\alpha| \geq 1$ for each $A \rightarrow \alpha \in R$).

A *Probabilistic Context-Free Grammar* (PCFG) is a quintuple $(N, W, R, S, \boldsymbol{\theta})$ where N, W, R and S are the nonterminals, terminals, rules and start symbol of a CFG respectively, and $\boldsymbol{\theta}$ is a vector of non-negative reals indexed by R that satisfy $\sum_{\alpha \in R_A} \theta_{A \rightarrow \alpha} = 1$ for each $A \in N$, where $R_A = \{A \rightarrow \alpha : A \rightarrow \alpha \in R\}$ is the set of rules expanding A.

Informally, $\theta_{A \rightarrow \alpha}$ is the probability of a node labelled A expanding to a sequence of nodes labelled α, and the probability of a tree is the product of the probabilities of the rules used to construct each non-leaf node in it. More precisely, for each $X \in N \cup W$ a PCFG associates distributions G_X over the set of trees $\mathcal{T}_X$ generated by X as follows:

If $X \in W$ (i.e., if X is a terminal) then G_X is the distribution that puts probability 1 on the single-node tree labelled X.

If $X \in N$ (i.e., if X is a nonterminal) then:

$$G_X = \sum_{X \rightarrow B_1 \ldots B_n \in R_X} \theta_{X \rightarrow B_1 \ldots B_n} \mathrm{TD}_X(G_{B_1}, \ldots, G_{B_n}) \quad (1)$$

where R_X is the subset of rules in R expanding nonterminal $X \in N$, and:

$$\mathrm{TD}_X(G_1, \ldots, G_n) \left(\begin{array}{c} X \\ \wedge \\ t_1 \ldots t_n \end{array} \right) = \prod_{i=1}^{n} G_i(t_i).$$

That is, $\mathrm{TD}_X(G_1, \ldots, G_n)$ is a distribution over the set of trees $\mathcal{T}_X$ generated by nonterminal X, where each subtree t_i is generated independently from G_i. The PCFG generates the distribution G_S over the set of trees $\mathcal{T}_S$ generated by the start symbol S; the distribution over the strings it generates is obtained by marginalising over the trees.

In a Bayesian PCFG one puts Dirichlet priors $\mathrm{Dir}(\boldsymbol{\alpha})$ on the rule probability vector $\boldsymbol{\theta}$, such that there is one Dirichlet parameter $\alpha_{A \rightarrow \alpha}$ for each rule $A \rightarrow \alpha \in R$. There are Markov Chain Monte Carlo (MCMC) and Variational Bayes procedures for estimating the posterior distribution over rule probabilities $\boldsymbol{\theta}$ and parse trees given data consisting of terminal strings alone (Kurihara and Sato, 2006; Johnson et al., 2007a).

PCFGs can be viewed as recursive mixture models over trees. While PCFGs are expressive enough to describe a range of linguistically-interesting phenomena, PCFGs are *parametric models*, which limits their ability to describe phenomena where the set of basic units, as well as their properties, are the target of learning. Lexical acqusition is an example of a phenomenon that is naturally viewed as *non-parametric inference*, where the number of lexical entries (i.e., words) as well as their properties must be learnt from the data.

It turns out there is a straight-forward modification to the PCFG distribution (1) that makes it suitably non-parametric. As Johnson et al. (2007b) explain, by inserting a Dirichlet Process (DP) or Pitman-Yor Process (PYP) into the generative mechanism (1) the model "concentrates" mass on a subset of trees (Teh et al., 2006). Specifically, an Adaptor Grammar identifies a subset $A \subseteq N$ of *adapted nonterminals*. In an Adaptor Grammar the unadapted nonterminals $N \setminus A$ expand via (1), just as in a PCFG, but the distributions of the adapted nonterminals A are "concentrated" by passing them through a DP or PYP:

$$\begin{aligned} H_X &= \sum_{X \rightarrow B_1 \ldots B_n \in R_X} \theta_{X \rightarrow B_1 \ldots B_n} \mathrm{TD}_X(G_{B_1}, \ldots, G_{B_n}) \\ G_X &= \mathrm{PYP}(H_X, a_X, b_X) \end{aligned}$$

Here a_X and b_X are parameters of the PYP associated with the adapted nonterminal X. As Goldwater et al. (2011) explain, such Pitman-Yor Processes naturally generate power-law distributed data.

Informally, Adaptor Grammars can be viewed as caching entire subtrees of the adapted nonterminals. Roughly speaking, the probability of generating a particular subtree of an adapted nonterminal is proportional to the number of times that subtree has been generated before. This "rich get

richer" behaviour causes the distribution of subtrees to follow a power-law (the power is specified by the a_X parameter of the PYP). The PCFG rules expanding an adapted nonterminal X define the "base distribution" of the associated DP or PYP, and the a_X and b_X parameters determine how much mass is reserved for "new" trees.

There are several different procedures for inferring the parse trees and the rule probabilities given a corpus of strings: Johnson et al. (2007b) describe a MCMC sampler and Cohen et al. (2010) describe a Variational Bayes procedure. We use the MCMC procedure here since this has been successfully applied to word segmentation problems in previous work (Johnson, 2008).

2 Word segmentation with Adaptor Grammars

Perhaps the simplest word segmentation model is the *unigram model*, where utterances are modeled as sequences of words, and where each word is a sequence of segments (Brent, 1999; Goldwater et al., 2009). A unigram model can be expressed as an Adaptor Grammar with one adapted nonterminal Word (we indicate adapted nonterminals by underlining them in grammars here; regular expressions are expanded into right-branching productions).

$$\mathsf{Sentence} \rightarrow \mathsf{Word}^{+} \quad (2)$$

$$\underline{\mathsf{Word}} \rightarrow \mathsf{Phone}^{+} \quad (3)$$

The first rule (2) says that a sentence consists of one or more Words, while the second rule (3) states that a Word consists of a sequence of one or more Phones; we assume that there are rules expanding Phone into all possible phones. Because Word is an adapted nonterminal, the adaptor grammar memoises Word subtrees, which corresponds to learning the phone sequences for the words of the language.

The more sophisticated Adaptor Grammars discussed below can be understood as specialising either the first or the second of the rules in (2–3). The next two subsections review the Adaptor Grammar word segmentation models presented in Johnson (2008) and Johnson and Goldwater (2009): section 2.1 reviews how phonotactic syllable-structure constraints can be expressed with Adaptor Grammars, while section 2.2 reviews how phrase-like units called "collocations" capture inter-word dependencies. Section 2.3 presents the major novel contribution of this paper by explaining how we modify these adaptor grammars to capture some of the special properties of function words.

2.1 Syllable structure and phonotactics

The rule (3) models words as sequences of independently generated phones: this is what Goldwater et al. (2009) called the "monkey model" of word generation (it instantiates the metaphor that word types are generated by a monkey randomly banging on the keys of a typewriter). However, the words of a language are typically composed of one or more syllables, and explicitly modelling the internal structure of words typically improves word segmentation considerably.

Johnson (2008) suggested replacing (3) with the following model of word structure:

$$\underline{\mathsf{Word}} \rightarrow \mathsf{Syllable}^{1:4} \quad (4)$$

$$\mathsf{Syllable} \rightarrow (\mathsf{Onset})\ \mathsf{Rhyme} \quad (5)$$

$$\underline{\mathsf{Onset}} \rightarrow \mathsf{Consonant}^{+} \quad (6)$$

$$\mathsf{Rhyme} \rightarrow \mathsf{Nucleus}\ (\mathsf{Coda}) \quad (7)$$

$$\underline{\mathsf{Nucleus}} \rightarrow \mathsf{Vowel}^{+} \quad (8)$$

$$\underline{\mathsf{Coda}} \rightarrow \mathsf{Consonant}^{+} \quad (9)$$

Here and below superscripts indicate iteration (e.g., a Word consists of 1 to 4 Syllables), while an Onset consists of an unbounded number of Consonants), while parentheses indicate optionality (e.g., a Rhyme consists of an obligatory Nucleus followed by an optional Coda). We assume that there are rules expanding Consonant and Vowel to the set of all consonants and vowels respectively (this amounts to assuming that the learner can distinguish consonants from vowels). Because Onset, Nucleus and Coda are adapted, this model learns the possible syllable onsets, nucleii and coda of the language, even though neither syllable structure nor word boundaries are explicitly indicated in the input to the model.

The model just described assumes that word-internal syllables have the same structure as word-peripheral syllables, but in languages such as English word-peripheral onsets and codas can be more complex than the corresponding word-internal onsets and codas. For example, the word "string" begins with the onset cluster *str*, which is relatively rare word-internally. Johnson (2008) showed that word segmentation accuracy improves if the model can learn different consonant sequences for word-inital onsets and word-final codas. It is easy to express this as an Adaptor

Grammar: (4) is replaced with (10–11) and (12–17) are added to the grammar.

$$\text{Word} \rightarrow \text{SyllableIF} \quad (10)$$
$$\text{Word} \rightarrow \text{SyllableI Syllable}^{0:2}\ \text{SyllableF} \quad (11)$$
$$\text{SyllableIF} \rightarrow (\text{OnsetI})\ \text{RhymeF} \quad (12)$$
$$\text{SyllableI} \rightarrow (\text{OnsetI})\ \text{Rhyme} \quad (13)$$
$$\text{SyllableF} \rightarrow (\text{Onset})\ \text{RhymeF} \quad (14)$$
$$\text{OnsetI} \rightarrow \text{Consonant}^{+} \quad (15)$$
$$\text{RhymeF} \rightarrow \text{Nucleus}\ (\text{CodaF}) \quad (16)$$
$$\text{CodaF} \rightarrow \text{Consonant}^{+} \quad (17)$$

In this grammar the suffix "I" indicates a word-initial element, and "F" indicates a word-final element. Note that the model simply has the ability to learn that different clusters can occur word-peripherally and word-internally; it is not given any information about the relative complexity of these clusters.

2.2 Collocation models of inter-word dependencies

Goldwater et al. (2009) point out the detrimental effect that inter-word dependencies can have on word segmentation models that assume that the words of an utterance are independently generated. Informally, a model that generates words independently is likely to incorrectly segment multi-word expressions such as "the doggie" as single words because the model has no way to capture word-to-word dependencies, e.g., that "doggie" is typically preceded by "the". Goldwater et al show that word segmentation accuracy improves when the model is extended to capture bigram dependencies.

Adaptor grammar models cannot express bigram dependencies, but they can capture similiar inter-word dependencies using phrase-like units that Johnson (2008) calls collocations. Johnson and Goldwater (2009) showed that word segmentation accuracy improves further if the model learns a nested hierarchy of collocations. This can be achieved by replacing (2) with (18–21).

$$\text{Sentence} \rightarrow \text{Colloc3}^{+} \quad (18)$$
$$\text{Colloc3} \rightarrow \text{Colloc2}^{+} \quad (19)$$
$$\text{Colloc2} \rightarrow \text{Colloc1}^{+} \quad (20)$$
$$\text{Colloc1} \rightarrow \text{Word}^{+} \quad (21)$$

Informally, Colloc1, Colloc2 and Colloc3 define a nested hierarchy of phrase-like units. While not designed to correspond to syntactic phrases, by examining the sample parses induced by the Adaptor Grammar we noticed that the collocations often correspond to noun phrases, prepositional phrases or verb phrases. This motivates the extension to the Adaptor Grammar discussed below.

2.3 Incorporating "function words" into collocation models

The starting point and baseline for our extension is the adaptor grammar with syllable structure phonotactic constraints and three levels of collocational structure (5-21), as prior work has found that this yields the highest word segmentation token f-score (Johnson and Goldwater, 2009).

Our extension assumes that the Colloc1 – Colloc3 constituents are in fact phrase-like, so we extend the rules (19–21) to permit an optional sequence of monosyllabic words at the left edge of each of these constituents. Our model thus captures two of the properties of function words discussed in section 1.1: they are monosyllabic (and thus phonologically simple), and they appear on the periphery of phrases. (We put "function words" in scare quotes below because our model only approximately captures the linguistic properties of function words).

Specifically, we replace rules (19–21) with the following sequence of rules:

$$\text{Colloc3} \rightarrow (\text{FuncWords3})\ \text{Colloc2}^{+} \quad (22)$$
$$\text{Colloc2} \rightarrow (\text{FuncWords2})\ \text{Colloc1}^{+} \quad (23)$$
$$\text{Colloc1} \rightarrow (\text{FuncWords1})\ \text{Word}^{+} \quad (24)$$
$$\text{FuncWords3} \rightarrow \text{FuncWord3}^{+} \quad (25)$$
$$\text{FuncWord3} \rightarrow \text{SyllableIF} \quad (26)$$
$$\text{FuncWords2} \rightarrow \text{FuncWord2}^{+} \quad (27)$$
$$\text{FuncWord2} \rightarrow \text{SyllableIF} \quad (28)$$
$$\text{FuncWords1} \rightarrow \text{FuncWord1}^{+} \quad (29)$$
$$\text{FuncWord1} \rightarrow \text{SyllableIF} \quad (30)$$

This model memoises (i.e., learns) both the individual "function words" and the sequences of "function words" that modify the Colloc1 – Colloc3 constituents. Note also that "function words" expand directly to SyllableIF, which in turn expands to a monosyllable with a word-initial onset and word-final coda. This means that "function words" are memoised independently of the "content words" that Word expands to; i.e., the model learns distinct "function word" and "content word" vocabularies. Figure 1 depicts a sample parse generated by this grammar.

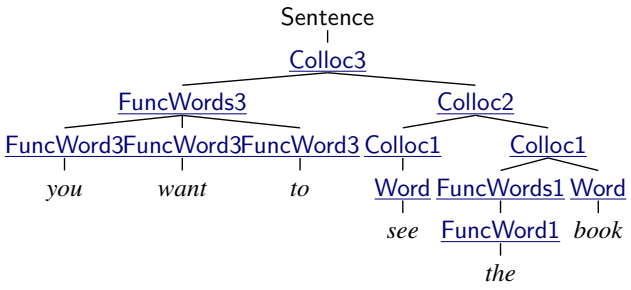


Figure 1: A sample parse generated by the "function word" Adaptor Grammar with rules (10–18) and (22–30). To simplify the parse we only show the root node and the adapted nonterminals, and replace word-internal structure by the word's orthographic form.

This grammar builds in the fact that function words appear on the left periphery of phrases. This is true of languages such as English, but is not true cross-linguistically. For comparison purposes we also include results for a mirror-image model that permits "function words" on the right periphery, a model which permits "function words" on both the left and right periphery (achieved by changing rules 22–24), as well as a model that analyses all words as monosyllabic.

Section 4 explains how a learner could use Bayesian model selection to determine that function words appear on the left periphery in English by comparing the posterior probability of the data under our "function word" Adaptor Grammar to that obtained using a grammar which is identical except that rules (22–24) are replaced with the mirror-image rules in which "function words" are attached to the right periphery.

3 Word segmentation results

This section presents results of running our Adaptor Grammar models on subsets of the Bernstein-Ratner (1987) corpus of child-directed English. We use the Adaptor Grammar software available from http://web.science.mq.edu.au/~mjohnson/ with the same settings as described in Johnson and Goldwater (2009), i.e., we perform Bayesian inference with "vague" priors for all hyperparameters (so there are no adjustable parameters in our models), and perform 8 different MCMC runs of each condition with table-label resampling for 2,000 sweeps of the training data. At every 10th sweep of the last 1,000 sweeps we use the model to segment the entire corpus (even if it is only trained on a subset of it), so we collect 800 sample segmentations of each utterance. The most frequent segmentation in these 800 sample segmentations is the one we score in the evaluations below.

Model	Token f-score	Boundary precision	Boundary recall
Baseline	0.872	0.918	0.956
+ left FWs	**0.924**	0.935	**0.990**
+ left + right FWs	0.912	**0.957**	0.953

Table 1: Mean token f-scores and boundary precision and recall results averaged over 8 trials, each consisting of 8 MCMC runs of models trained and tested on the full Bernstein-Ratner (1987) corpus (the standard deviations of all values are less than 0.006; Wilcox sign tests show the means of all token f-scores differ $p < 2\text{e-}4$).

3.1 Word segmentation with "function word" models

Here we evaluate the word segmentations found by the "function word" Adaptor Grammar model described in section 2.3 and compare it to the baseline grammar with collocations and phonotactics from Johnson and Goldwater (2009). Figure 2 presents the standard token and lexicon (i.e., type) f-score evaluations for word segmentations proposed by these models (Brent, 1999), and Table 1 summarises the token and lexicon f-scores for the major models discussed in this paper. It is interesting to note that adding "function words" improves token f-score by more than 4%, corresponding to a 40% reduction in overall error rate.

When the training data is very small the Monosyllabic grammar produces the highest accuracy results, presumably because a large proportion of the words in child-directed speech are monosyllabic. However, at around 25 sentences the more complex models that are capable of finding multi-syllabic words start to become more accurate.

It's interesting that after about 1,000 sentences the model that allows "function words" only on the right periphery is considerably less accurate than the baseline model. Presumably this is because it tends to misanalyse multi-syllabic words on the right periphery as sequences of monosyllabic words.

The model that allows "function words" only on the left periphery is more accurate than the model that allows them on both the left and right periphery when the input data ranges from about 100 to about 1,000 sentences, but when the training data

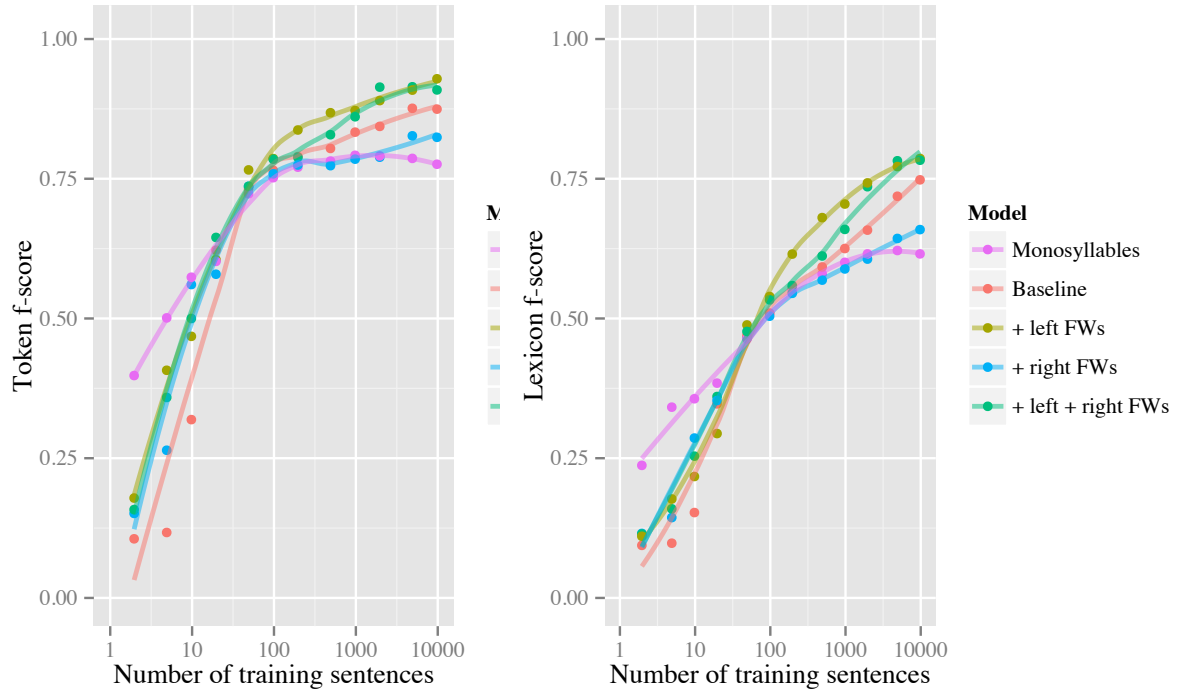


Figure 2: Token and lexicon (i.e., type) f-score on the Bernstein-Ratner (1987) corpus as a function of training data size for the baseline model, the model where "function words" can appear on the left periphery, a model where "function words" can appear on the right periphery, and a model where "function words" can appear on both the left and the right periphery. For comparison purposes we also include results for a model that assumes that all words are monosyllabic.

is larger than about 1,000 sentences both models are equally accurate.

3.2 Content and function words found by "function word" model

As noted earlier, the "function word" model generates function words via adapted nonterminals other than the Word category. In order to better understand just how the model works, we give the 5 most frequent words in each word category found during 8 MCMC runs of the left-peripheral "function word" grammar above:

Word : *book, doggy, house, want, I*
FuncWord1 : *a, the, your, little*[1], *in*
FuncWord2 : *to, in, you, what, put*
FuncWord3 : *you, a, what, no, can*

Interestingly, these categories seem fairly reasonable. The Word category includes open-class nouns and verbs, the FuncWord1 category includes noun modifiers such as determiners, while the FuncWord2 and FuncWord3 categories include prepositions, pronouns and auxiliary verbs.

[1]The phone 'l' is generated by both Consonant and Vowel, so "little" can be (incorrectly) analysed as one syllable.

Thus, the present model, initially aimed at segmenting words from continuous speech, shows three interesting characteristics that are also exhibited by human infants: it distinguishes between function words and content words (Shi and Werker, 2001), it allows learners to acquire at least some of the function words of their language (e.g. (Shi et al., 2006)); and furthermore, it may also allow them to start grouping together function words according to their category (Cauvet et al., 2014; Shi and Melançon, 2010).

4 Are "function words" on the left or right periphery?

We have shown that a model that expects function words on the left periphery performs more accurate word segmentation on English, where function words do indeed typically occur on the left periphery, leaving open the question: how could a learner determine whether function words generally appear on the left or the right periphery of phrases in the language they are learning? This question is important because knowing the side where function words preferentially occur is re-

lated to the question of the direction of syntactic headedness in the language, and an accurate method for identifying the location of function words might be useful for initialising a syntactic learner. Experimental evidence suggests that infants as young as 8 months of age already expect function words on the correct side for their language — left-periphery for Italian infants and right-periphery for Japanese infants (Gervain et al., 2008) — so it is interesting to see whether purely distributional learners such as the ones studied here can identify the correct location of function words in phrases.

We experimented with a variety of approaches that use a single adaptor grammar inference process, but none of these were successful. For example, we hoped that given an Adaptor Grammar that permits "function words" on both the left and right periphery, the inference procedure would decide that the right-periphery rules simply are not used in a language like English. Unfortunately we did not find this in our experiments; the right-periphery rules were used almost as often as the left-periphery rules (recall that a large fraction of the words in English child-directed speech are monosyllabic).

In this section, we show that learners could use Bayesian model selection to determine that function words appear on the left periphery in English by comparing the marginal probability of the data for the left-periphery and the right-periphery models.

Instead, we used Bayesian model selection techniques to determine whether left-peripheral or a right-peripheral model better fits the unsegmented utterances that constitute the training data.[2] While Bayesian model selection is in principle straight-forward, it turns out to require the ratio of two integrals (for the "evidence" or marginal likelihood) that are often intractable to compute.

Specifically, given a training corpus D of unsegmented sentences and model families G_1 and G_2 (here the "function word" adaptor grammars with left-peripheral and right-peripheral attachment respectively), the Bayes factor K is the ratio of the marginal likelihoods of the data:

$$K = \frac{\mathrm{P}(D \mid G_1)}{\mathrm{P}(D \mid G_2)}$$

[2]Note that neither the left-peripheral nor the right-peripheral model is correct: even strongly left-headed languages like English typically contain a few right-headed constructions. For example, "ago" is arguably the head of the phrase "ten years ago".

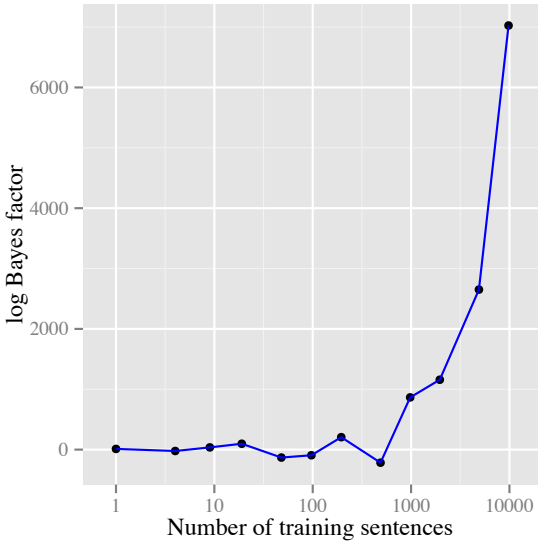


Figure 3: Bayes factor in favour of left-peripheral "function word" attachment as a function of the number of sentences in the training corpus, calculated using the Harmonic Mean estimator (see warning in text).

where the marginal likelihood or "evidence" for a model G is obtained by integrating over all of the hidden or latent structure and parameters $\boldsymbol{\theta}$:

$$\mathrm{P}(D \mid G) = \int_{\Delta} \mathrm{P}(D, \boldsymbol{\theta} \mid G)\, d\boldsymbol{\theta} \quad (31)$$

Here the variable $\boldsymbol{\theta}$ ranges over the space Δ of all possible parses for the utterances in D and all possible configurations of the Pitman-Yor processes and their parameters that constitute the "state" of the Adaptor Grammar G. While the probability of any specific Adaptor Grammar configuration $\boldsymbol{\theta}$ is not too hard to calculate (the MCMC sampler for Adaptor Grammars can print this after each sweep through D), the integral in (31) is in general intractable.

Textbooks such as Murphy (2012) describe a number of methods for calculating $\mathrm{P}(D \mid G)$, but most of them assume that the parameter space Δ is continuous and so cannot be directly applied here. The Harmonic Mean estimator (32) for (31), which we used here, is a popular estimator for (31) because it only requires the ability to calculate $\mathrm{P}(D, \boldsymbol{\theta} \mid G)$ for samples from $\mathrm{P}(\boldsymbol{\theta} \mid D, G)$:

$$\mathrm{P}(D \mid G) \approx \left(\frac{1}{n} \sum_{i=1}^{n} \frac{1}{\mathrm{P}(D, \boldsymbol{\theta}_i \mid G)} \right)^{-1}$$

where $\boldsymbol{\theta}_i, \ldots, \boldsymbol{\theta}_n$ are n samples from $\mathrm{P}(\boldsymbol{\theta} \mid$

D, G), which can be generated by the MCMC procedure.

Figure 3 depicts how the Bayes factor in favour of left-peripheral attachment of "function words" varies as a function of the number of utterances in the training data D (calculated from the last 1000 sweeps of 8 MCMC runs of the corresponding adaptor grammars). As that figure shows, once the training data contains more than about 1,000 sentences the evidence for the left-peripheral grammar becomes very strong. On the full training data the estimated log Bayes factor is over 6,000, which would constitute overwhelming evidence in favour of left-peripheral attachment.

Unfortunately, as Murphy and others warn, the Harmonic Mean estimator is extremely unstable (Radford Neal calls it "the worst MCMC method ever" in his blog), so we think it is important to confirm these results using a more stable estimator. However, given the magnitude of the differences and the fact that the two models being compared are of similar complexity, we believe that these results suggest that Bayesian model selection can be used to determine properties of the language being learned.

5 Conclusions and future work

This paper showed that the word segmentation accuracy of a state-of-the-art Adaptor Grammar model is significantly improved by extending it so that it explicitly models some properties of function words. We also showed how Bayesian model selection can be used to identify that function words appear on the left periphery of phrases in English, even though the input to the model only consists of an unsegmented sequence of phones.

Of course this work only scratches the surface in terms of investigating the role of function words in language acquisition. It would clearly be very interesting to examine the performance of these models on other corpora of child-directed English, as well as on corpora of child-directed speech in other languages. Our evaluation focused on word-segmentation, but we could also evaluate the effect that modelling "function words" has on other aspects of the model, such as its ability to learn syllable structure.

The models of "function words" we investigated here only capture two of the 7 linguistic properties of function words identified in section 1 (i.e., that function words tend to be monosyllabic, and that they tend to appear phrase-peripherally), so it would be interesting to develop and explore models that capture other linguistic properties of function words. For example, following the suggestion by Hochmann et al. (2010) that human learners use frequency cues to identify function words, it might be interesting to develop computational models that do the same thing. In an Adaptor Grammar the frequency distribution of function words might be modelled by specifying the prior for the Pitman-Yor Process parameters associated with the function words' adapted nonterminals so that it prefers to generate a small number of high-frequency items.

It should also be possible to develop models which capture the fact that function words tend not to be topic-specific. Johnson et al. (2010) and Johnson et al. (2012) show how Adaptor Grammars can model the association between words and non-linguistic "topics"; perhaps these models could be extended to capture some of the semantic properties of function words.

It would also be interesting to further explore the extent to which Bayesian model selection is a useful approach to linguistic "parameter setting". In order to do this it is imperative to develop better methods than the problematic "Harmonic Mean" estimator used here for calculating the evidence (i.e., the marginal probability of the data) that can handle the combination of discrete and continuous hidden structure that occur in computational linguistic models.

As well as substantially improving the accuracy of unsupervised word segmentation, this work is interesting because it suggests a connection between unsupervised word segmentation and the induction of syntactic structure. It is reasonable to expect that hierarchical non-parametric Bayesian models such as Adaptor Grammars may be useful tools for exploring such a connection.

Acknowledgments

This work was supported in part by the Australian Research Council's Discovery Projects funding scheme (project numbers DP110102506 and DP110102593), the European Research Council (ERC-2011-AdG-295810 BOOTPHON), the Agence Nationale pour la Recherche (ANR-10-LABX-0087 IEC, and ANR-10-IDEX-0001-02 PSL*), and the Mairie de Paris, Ecole des Hautes Etudes en Sciences Sociales, the Ecole Normale Supérieure, and the Fondation Pierre Gilles de Gennes.

References

N. Bernstein-Ratner. 1987. The phonology of parent-child speech. In K. Nelson and A. van Kleeck, editors, *Children's Language*, volume 6, pages 159–174. Erlbaum, Hillsdale, NJ.

M. Brent. 1999. An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*, 34:71–105.

E. Cauvet, R. Limissuri, S. Millotte, K. Skoruppa, D. Cabrol, and A. Christophe. 2014. Function words constrain on-line recognition of verbs and nouns in French 18-month-olds. *Language Learning and Development*, pages 1–18.

A. Christophe, S. Millotte, S. Bernal, and J. Lidz. 2008. Bootstrapping lexical and syntactic acquisition. *Language and Speech*, 51(1-2):61–75.

S. B. Cohen, D. M. Blei, and N. A. Smith. 2010. Variational inference for adaptor grammars. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 564–572, Los Angeles, California, June. Association for Computational Linguistics.

K. Demuth and E. McCullough. 2009. The prosodic (re-)organization of childrens early English articles. *Journal of Child Language*, 36(1):173–200.

J. Elman, E. Bates, M. H. Johnson, A. Karmiloff-Smith, D. Parisi, and K. Plunkett. 1996. *Rethinking Innateness: A Connectionist Perspective on Development*. MIT Press/Bradford Books, Cambridge, MA.

V. Fromkin, editor. 2001. *Linguistics: An Introduction to Linguistic Theory*. Blackwell, Oxford, UK.

J. Gervain, M. Nespor, R. Mazuka, R. Horie, and J. Mehler. 2008. Bootstrapping word order in prelexical infants: A japaneseitalian cross-linguistic study. *Cognitive Psychology*, 57(1):56 – 74.

S. Goldwater, T. L. Griffiths, and M. Johnson. 2009. A Bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, 112(1):21–54.

S. Goldwater, T. L. Griffiths, and M. Johnson. 2011. Producing power-law distributions and damping word frequencies with two-stage language models. *Journal of Machine Learning Research*, 12:2335–2382.

P. A. Hallé, C. Durand, and B. de Boysson-Bardies. 2008. Do 11-month-old French infants process articles? *Language and Speech*, 51(1-2):23–44.

J.-R. Hochmann, A. D. Endress, and J. Mehler. 2010. Word frequency as a cue for identifying function words in infancy. *Cognition*, 115(3):444 – 457.

M. Johnson and S. Goldwater. 2009. Improving nonparameteric Bayesian inference: experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 317–325, Boulder, Colorado, June. Association for Computational Linguistics.

M. Johnson, T. Griffiths, and S. Goldwater. 2007a. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 139–146, Rochester, New York. Association for Computational Linguistics.

M. Johnson, T. L. Griffiths, and S. Goldwater. 2007b. Adaptor Grammars: A framework for specifying compositional nonparametric Bayesian models. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 641–648. MIT Press, Cambridge, MA.

M. Johnson, K. Demuth, M. Frank, and B. Jones. 2010. Synergies in learning words and their referents. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1018–1026.

M. Johnson, K. Demuth, and M. Frank. 2012. Exploiting social information in grounded language learning via grammatical reduction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 883–891, Jeju Island, Korea, July. Association for Computational Linguistics.

M. Johnson. 2008. Using Adaptor Grammars to identify synergies in the unsupervised acquisition of linguistic structure. In *Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics*, pages 398–406, Columbus, Ohio. Association for Computational Linguistics.

Y. Kedar, M. Casasola, and B. Lust. 2006. Getting there faster: 18- and 24-month-old infants' use of function words to determine reference. *Child Development*, 77(2):325–338.

K. Kurihara and T. Sato. 2006. Variational Bayesian grammar induction for natural language. In Y. Sakakibara, S. Kobayashi, K. Sato, T. Nishino, and E. Tomita, editors, *Grammatical Inference: Algorithms and Applications*, pages 84–96. Springer.

K. P. Murphy. 2012. *Machine learning: a probabilistic perspective*. The MIT Press.

V. L. Shafer, D. W. Shucard, J. L. Shucard, and L. Gerken. 1998. An electrophysiological study of infants' sensitivity to the sound patterns of English

speech. *Journal of Speech, Language and Hearing Research*, 41(4):874.

R. Shi and M. Lepage. 2008. The effect of functional morphemes on word segmentation in preverbal infants. *Developmental Science*, 11(3):407–413.

R. Shi and A. Melançon. 2010. Syntactic categorization in French-learning infants. *Infancy*, 15(517–533).

R. Shi and J. Werker. 2001. Six-months old infants' preference for lexical words. *Psychological Science*, 12:71–76.

R. Shi, A. Cutler, J. Werker, and M. Cruickshank. 2006. Frequency and form as determinants of functor sensitivity in English-acquiring infants. *The Journal of the Acoustical Society of America*, 119(6):EL61–EL67.

Y. W. Teh, M. Jordan, M. Beal, and D. Blei. 2006. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101:1566–1581.

R. Zangl and A. Fernald. 2007. Increasing flexibility in children's online processing of grammatical and nonce determiners in fluent speech. *Language Learning and Development*, 3(3):199–231.

Max-Margin Tensor Neural Network for Chinese Word Segmentation

Wenzhe Pei **Tao Ge** **Baobao Chang***
Key Laboratory of Computational Linguistics, Ministry of Education
School of Electronics Engineering and Computer Science, Peking University
Beijing, P.R.China, 100871
{peiwenzhe,getao,chbb}@pku.edu.cn

Abstract

Recently, neural network models for natural language processing tasks have been increasingly focused on for their ability to alleviate the burden of manual feature engineering. In this paper, we propose a novel neural network model for Chinese word segmentation called Max-Margin Tensor Neural Network (**MMTNN**). By exploiting tag embeddings and tensor-based transformation, **MMTNN** has the ability to model complicated interactions between tags and context characters. Furthermore, a new tensor factorization approach is proposed to speed up the model and avoid overfitting. Experiments on the benchmark dataset show that our model achieves better performances than previous neural network models and that our model can achieve a competitive performance with minimal feature engineering. Despite Chinese word segmentation being a specific case, **MMTNN** can be easily generalized and applied to other sequence labeling tasks.

1 Introduction

Unlike English and other western languages, Chinese do not delimit words by white-space. Therefore, word segmentation is a preliminary and important pre-process for Chinese language processing. Most previous systems address this problem by treating this task as a sequence labeling problem where each character is assigned a tag indicating its position in the word. These systems are effective because researchers can incorporate a large body of handcrafted features into the models. However, the ability of these models is restricted by the design of features and the number of features could be so large that the result models are too large for practical use and prone to overfit on training corpus.

Recently, neural network models have been increasingly focused on for their ability to minimize the effort in feature engineering. Collobert et al. (2011) developed the SENNA system that approaches or surpasses the state-of-the-art systems on a variety of sequence labeling tasks for English. Zheng et al. (2013) applied the architecture of Collobert et al. (2011) to Chinese word segmentation and POS tagging and proposed a perceptron-style algorithm to speed up the training process with negligible loss in performance.

Workable as previous neural network models seem, a limitation of them to be pointed out is that the tag-tag interaction, tag-character interaction and character-character interaction are not well modeled. In conventional feature-based linear (log-linear) models, these interactions are explicitly modeled as features. Take phrase "打篮球(play basketball)" as an example, assuming we are labeling character C_0="篮", possible features could be:

$$f_1 = \begin{cases} 1 & C_{-1}\text{="打" and } C_1\text{="球" and } y_0\text{="B"} \\ 0 & \text{else} \end{cases}$$

$$f_2 = \begin{cases} 1 & C_0\text{="篮" and } y_0\text{="B" and } y_{-1}\text{="S"} \\ 0 & \text{else} \end{cases}$$

To capture more interactions, researchers have designed a large number of features based on linguistic intuition and statistical information. In previous neural network models, however, hardly can such interactional effects be fully captured relying only on the simple transition score and the single non-linear transformation (See section 2). In order to address this problem, we propose a new model called Max-Margin Tensor Neural Network (**MMTNN**) that explicitly models the interactions

*Corresponding author

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 293–303,
Baltimore, Maryland, USA, June 23-25 2014.

between tags and context characters by exploiting tag embeddings and tensor-based transformation. Moreover, we propose a tensor factorization approach that effectively improves the model efficiency and prevents from overfitting. We evaluate the performance of Chinese word segmentation on the PKU and MSRA benchmark datasets in the second International Chinese Word Segmentation Bakeoff (Emerson, 2005) which are commonly used for evaluation of Chinese word segmentation. Experiment results show that our model outperforms other neural network models.

Although we focus on the question that how far we can go without using feature engineering in this paper, the study of deep learning for NLP tasks is still a new area in which it is currently challenging to surpass the state-of-the-art without additional features. Following Mansur et al. (2013), we wonder how well our model can perform with minimal feature engineering. Therefore, we integrate additional simple character bigram features into our model and the result shows that our model can achieve a competitive performance that other systems hardly achieve unless they use more complex task-specific features.

The main contributions of our work are as follows:

- We propose a Max-Margin Tensor Neural Network for Chinese word segmentation without feature engineering. The test results on the benchmark dataset show that our model outperforms previous neural network models.
- We propose a new tensor factorization approach that models each tensor slice as the product of two low-rank matrices. Not only does this approach improve the efficiency of our model but also it avoids the risk of overfitting.
- Compared with previous works that use a large number of handcrafted features, our model can achieve a competitive performance with minimal feature engineering.
- Despite Chinese word segmentation being a specific case, our approach can be easily generalized to other sequence labeling tasks.

The remaining part of this paper is organized as follows. Section 2 describes the details of conventional neural network architecture. Section 3 describes the details of our model. Experiment results are reported in Section 4. Section 5 reviews the related work. The conclusions are given in Section 6.

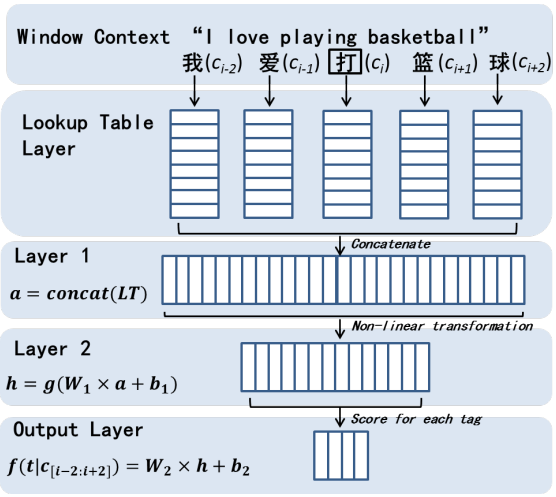


Figure 1: Conventional Neural Network

2 Conventional Neural Network

2.1 Lookup Table

The idea of distributed representation for symbolic data is one of the most important reasons why the neural network works. It was proposed by Hinton (1986) and has been a research hot spot for more than twenty years (Bengio et al., 2003; Collobert et al., 2011; Schwenk et al., 2012; Mikolov et al., 2013a). Formally, in the Chinese word segmentation task, we have a character dictionary D of size $|D|$. Unless otherwise specified, the character dictionary is extracted from the training set and unknown characters are mapped to a special symbol that is not used elsewhere. Each character $c \in D$ is represented as a real-valued vector (character embedding) $Embed(c) \in \mathbb{R}^d$ where d is the dimensionality of the vector space. The character embeddings are then stacked into a embedding matrix $M \in \mathbb{R}^{d \times |D|}$. For a character $c \in D$ that has an associated index k, the corresponding character embedding $Embed(c) \in \mathbb{R}^d$ is retrieved by the Lookup Table layer as shown in Figure 1:

$$Embed(c) = Me_k \tag{1}$$

Here $e_k \in \mathbb{R}^{|D|}$ is a binary vector which is zero in all positions except at k-th index. The Lookup Table layer can be seen as a simple projection layer where the character embedding for each context

character is achieved by table lookup operation according to their indices. The embedding matrix M is initialized with small random numbers and trained by back-propagation. We will analyze in more detail about the effect of character embeddings in Section 4.

2.2 Tag Scoring

The most common tagging approach is the window approach. The window approach assumes that the tag of a character largely depends on its neighboring characters. Given an input sentence $c_{[1:n]}$, a window of size w slides over the sentence from character c_1 to c_n. We set $w = 5$ in all experiments. As shown in Figure 1, at position $c_i, 1 \leq i \leq n$, the context characters are fed into the Lookup Table layer. The characters exceeding the sentence boundaries are mapped to one of two special symbols, namely "start" and "end" symbols. The character embeddings extracted by the Lookup Table layer are then concatenated into a single vector $a \in \mathbb{R}^{H_1}$, where $H_1 = w \cdot d$ is the size of Layer 1. Then a is fed into the next layer which performs linear transformation followed by an element-wise activation function g such as *tanh*, which is used in our experiments:

$$h = g(W_1 a + b_1) \tag{2}$$

where $W_1 \in \mathbb{R}^{H_2 \times H_1}$, $b_1 \in \mathbb{R}^{H_2 \times 1}$, $h \in \mathbb{R}^{H_2}$. H_2 is a hyper-parameter which is the number of hidden units in Layer 2. Given a set of tags T of size $|T|$, a similar linear transformation is performed except that no non-linear function is followed:

$$f(\boldsymbol{t}|c_{[i-2:i+2]}) = W_2 h + b_2 \tag{3}$$

where $W_2 \in \mathbb{R}^{|T| \times H_2}$, $b_2 \in \mathbb{R}^{|T| \times 1}$. $f(\boldsymbol{t}|c_{[i-2:i+2]}) \in \mathbb{R}^{|T|}$ is the score vector for each possible tag. In Chinese word segmentation, the most prevalent tag set T is BMES tag set, which uses 4 tags to carry word boundary information. It uses B, M, E and S to denote the Beginning, the Middle, the End of a word and a Single character forming a word respectively. We use this tag set in our method.

2.3 Model Training and Inference

Despite sharing commonalities mentioned above, previous work models the segmentation task differently and therefore uses different training and inference procedure. Mansur et al. (2013) modeled Chinese word segmentation as a series of classification task at each position of the sentence in which the tag score is transformed into probability using softmax function:

$$p(t_i|c_{[i-2:i+2]}) = \frac{exp(f(t_i|c_{[i-2:i+2]}))}{\sum_{t'} exp(f(t'|c_{[i-2:i+2]}))}$$

The model is then trained in MLE-style which maximizes the log-likelihood of the tagged data. Obviously, it is a local model which cannot capture the dependency between tags and does not support to infer the tag sequence globally.

To model the tag dependency, previous neural network models (Collobert et al., 2011; Zheng et al., 2013) introduce a transition score A_{ij} for jumping from tag $i \in T$ to tag $j \in T$. For a input sentence $c_{[1:n]}$ with a tag sequence $t_{[1:n]}$, a sentence-level score is then given by the sum of transition and network scores:

$$s(c_{[1:n]}, t_{[1:n]}, \theta) = \sum_{i=1}^{n} (A_{t_{i-1}t_i} + f_\theta(t_i|c_{[i-2:i+2]})) \tag{4}$$

where $f_\theta(t_i|c_{[i-2:i+2]})$ indicates the score output for tag t_i at the i-th character by the network with parameters $\theta = (M, A, W_1, b_1, W_2, b_2)$. Given the sentence-level score, Zheng et al. (2013) proposed a perceptron-style training algorithm inspired by the work of Collins (2002). Compared with Mansur et al. (2013), their model is a global one where the training and inference is performed at sentence-level.

Workable as these methods seem, one of the limitations of them is that the tag-tag interaction and the neural network are modeled seperately. The simple tag-tag transition neglects the impact of context characters and thus limits the ability to capture flexible interactions between tags and context characters. Moreover, the simple non-linear transformation in equation (2) is also poor to model the complex interactional effects in Chinese word segmentation.

3 Max-Margin Tensor Neural Network

3.1 Tag Embedding

To better model the tag-tag interaction given the context characters, distributed representation for tags instead of traditional discrete symbolic representation is used in our model. Similar to character embeddings, given a fixed-sized tag set T, the tag embeddings for tags are stored in a tag embedding matrix $L \in \mathbb{R}^{d \times |T|}$, where d is the dimensionality

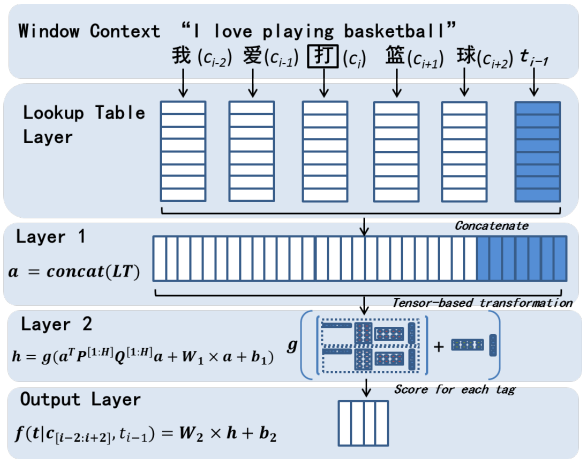


Figure 2: Max-Margin Tensor Neural Network

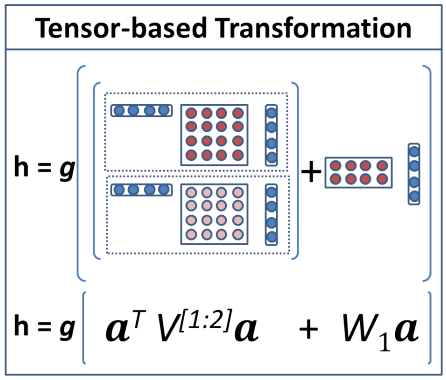


Figure 3: The tensor-based transformation in Layer 2. a is the input from Layer 1. V is the tensor parameter. Each dashed box represents one of the H_2-many tensor slices, which defines the bilinear form on vector a.

of the vector space (same with character embeddings). Then the tag embedding $Embed(t) \in \mathbb{R}^d$ for tag $t \in T$ with index k can be retrieved by the lookup operation:

$$Embed(t) = Le_k \tag{5}$$

where $e_k \in \mathbb{R}^{|T|\times 1}$ is a binary vector which is zero in all positions except at k-th index. The tag embeddings start from a random initialization and can be automatically trained by back-propagation. Figure 2 shows the new Lookup Table layer with tag embeddings. Assuming we are at the i-th character of a sentence, besides the character embeddings, the tag embeddings of the previous tags are also considered[1]. For a fast tag inference, only the previous tag t_{i-1} is used in our model even though a longer history of tags can be considered. The concatenation operation in Layer 1 then concatenates the character embeddings and tag embedding together into a long vector a. In this way, the tag representation can be directly incorporated in the neural network so that the tag-tag interaction and tag-character interaction can be explicitly modeled in deeper layers (See Section 3.2). Moreover, the transition score in equation (4) is not necessary in our model, because, by incorporating tag embedding into the neural network, the effect of tag-tag interaction and tag-character interaction are covered uniformly in one same model. Now equation (4) can be rewritten as follows:

$$s(c_{[1:n]}, t_{[1:n]}, \theta) = \sum_{i=1}^{n} f_\theta(t_i|c_{[i-2:i+2]}, t_{i-1}) \tag{6}$$

where $f_\theta(t_i|c_{[i-2:i+2]}, t_{i-1})$ is the score output for tag t_i at the i-th character by the network with parameters θ. Like Collobert et al. (2011) and Zheng et al. (2013), our model is also trained at sentence-level and carries out inference globally.

3.2 Tensor Neural Network

A tensor is a geometric object that describes relations between vectors, scalars, and other tensors. It can be represented as a multi-dimensional array of numerical values. An advantage of the tensor is that it can explicitly model multiple interactions in data. As a result, tensor-based model have been widely used in a variety of tasks (Salakhutdinov et al., 2007; Krizhevsky et al., 2010; Socher et al., 2013b).

In Chinese word segmentation, a proper modeling of the tag-tag interaction, tag-character interaction and character-character interaction is very important. In linear models, these kinds of interactions are usually modeled as features. In conventional neural network models, however, the input embeddings only implicitly interact through the non-linear function which can hardly model the complexity of the interactions. Given the advantage of tensors, we apply a tensor-based transformation to the input vector. Formally, we use a 3-way tensor $V^{[1:H_2]} \in \mathbb{R}^{H_2\times H_1\times H_1}$ to directly model the interactions, where H_2 is the size of

[1]We also tried the architecture in which the tag embedding of current tag is also considered, but this did not bring much improvement and runs slower

Layer 2 and $H_1 = (w+1) \cdot d$ is the size of concatenated vector a in Layer 1 as shown in Figure 2. Figure 3 gives an example of the tensor-based transformation[2]. The output of a tensor product is a vector $z \in \mathbb{R}^{H_2}$ where each dimension z_i is the result of the bilinear form defined by each tensor slice $V^{[i]} \in \mathbb{R}^{H_1 \times H_1}$:

$$z = a^T V^{[1:H_2]} a; z_i = a^T V^{[i]} a = \sum_{j,k} V^{[i]}_{jk} a_j a_k \tag{7}$$

Since vector a is the concatenation of character embeddings and the tag embedding, equation (7) can be written in the following form:

$$z_i = \sum_{p,q} \sum_{j,k} V^{[i]}_{(p,q,j,k)} E^{[p]}_j E^{[q]}_k$$

where $E^{[p]}_j$ is the j-th element of the p-th embedding in Lookup Table layer and $V^{[i]}_{(p,q,j,k)}$ is the corresponding coefficient for $E^{[p]}_j$ and $E^{[q]}_k$ in $V^{[i]}$. As we can see, in each tensor slice i, the embeddings are explicitly related in a bilinear form which captures the interactions between characters and tags. The multiplicative operations between tag embeddings and character embeddings can somehow be seen as "feature combination", which are hand-designed in feature-based models. Our model learns the information automatically and encodes them in tensor parameters and embeddings. Intuitively, we can interpret each slice of the tensor as capturing a specific type of tag-character interaction and character-character interaction.

Combining the tensor product with linear transformation, the tensor-based transformation in Layer 2 is defined as:

$$h = g(a^T V^{[1:H_2]} a + W_1 a + b_1) \tag{8}$$

where $W_1 \in \mathbb{R}^{H_2 \times H_1}$, $b_1 \in \mathbb{R}^{H_2 \times 1}$, $h \in \mathbb{R}^{H_2}$. In fact, equation (2) used in previous work is a special case of equation (8) when V is set to 0.

3.3 Tensor Factorization

Despite tensor-based transformation being effective for capturing the interactions, introducing tensor-based transformation into neural network models to solve sequence labeling task is time prohibitive since the tensor product operation drastically slows down the model. Without considering matrix optimization algorithms, the complexity of the non-linear transformation in equation (2) is $O(H_1 H_2)$ while the tensor operation complexity in equation (8) is $O(H_1^2 H_2)$. The tensor-based transformation is H_1 times slower. Moreover, the additional tensor could bring millions of parameters to the model which makes the model suffer from the risk of overfitting. To remedy this, we propose a tensor factorization approach that factorizes each tensor slice as the product of two low-rank matrices. Formally, each tensor slice $V^{[i]} \in \mathbb{R}^{H_1 \times H_1}$ is factorized into two low rank matrix $P^{[i]} \in \mathbb{R}^{H_1 \times r}$ and $Q^{[i]} \in \mathbb{R}^{r \times H_1}$:

$$V^{[i]} = P^{[i]} Q^{[i]}, 1 \leq i \leq H_2 \tag{9}$$

where $r \ll H_1$ is the number of factors. Substituting equation (9) into equation (8), we get the factorized tensor function:

$$h = g(a^T P^{[1:H_2]} Q^{[1:H_2]} a + W_1 a + b_1) \tag{10}$$

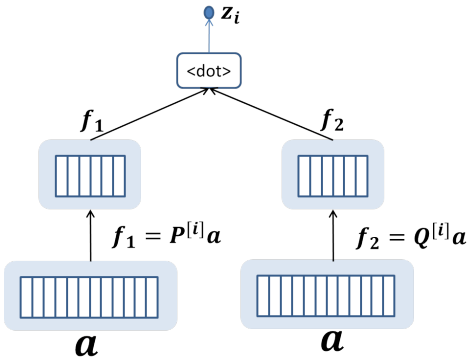


Figure 4: Tensor product with tensor factorization

Figure 4 illustrates the operation in each slice of the factorized tensor. First, vector a is projected into two r-dimension vectors f_1 and f_2. Then the output z_i for each tensor slice i is the dot-product of f_1 and f_2. The complexity of the tensor operation is now $O(rH_1H_2)$. As long as r is small enough, the factorized tensor operation would be much faster than the un-factorized one and the number of free parameters would also be much smaller, which prevent the model from overfitting.

3.4 Max-Margin Training

We use the Max-Margin criterion to train our model. Intuitively, the Max-Margin criterion provides an alternative to probabilistic, likelihood-based estimation methods by concentrating directly on the robustness of the decision boundary of a model (Taskar et al., 2005). We use $Y(x_i)$ to denote the set of all possible tag sequences for

[2]The bias term is omitted in Figure 3 for simplicity

a given sentence x_i and the correct tag sequence for x_i is y_i. The parameters of our model are $\theta = \{W_1, b_1, W_2, b_2, M, L, P^{[1:H_2]}, Q^{[1:H_2]}\}$. We first define a structured margin loss $\triangle(y_i, \hat{y})$ for predicting a tag sequence $\hat{y}$ for a given correct tag sequence y_i:

$$\triangle(y_i, \hat{y}) = \sum_{j}^{n} \kappa \mathbf{1}\{y_{i,j} \neq \hat{y}_j\} \qquad (11)$$

where n is the length of sentence x_i and κ is a discount parameter. The loss is proportional to the number of characters with an incorrect tag in the proposed tag sequence, which increases the more incorrect the proposed tag sequence is. For a given training instance (x_i, y_i), we search for the tag sequence with the highest score:

$$y^* = \arg\max_{\hat{y} \in Y(x)} s(x_i, \hat{y}, \theta) \qquad (12)$$

where the tag sequence is found and scored by the Tensor Neural Network via the function s in equation (6). The object of Max-Margin training is that the highest scoring tag sequence is the correct one: $y^* = y_i$ and its score will be larger up to a margin to other possible tag sequences $\hat{y} \in Y(x_i)$:

$$s(x, y_i, \theta) \geq s(x, \hat{y}, \theta) + \triangle(y_i, \hat{y})$$

This leads to the regularized objective function for m training examples:

$$\begin{aligned} J(\theta) &= \frac{1}{m}\sum_{i=1}^{m} l_i(\theta) + \frac{\lambda}{2}||\theta||^2 \\ l_i(\theta) &= \max_{\hat{y} \in Y(x_i)} (s(x_i, \hat{y}, \theta) + \triangle(y_i, \hat{y})) \\ &\quad -s(x_i, y_i, \theta)) \end{aligned} \qquad (13)$$

By minimizing this object, the score of the correct tag sequence y_i is increased and score of the highest scoring incorrect tag sequence $\hat{y}$ is decreased.

The objective function is not differentiable due to the hinge loss. We use a generalization of gradient descent called subgradient method (Ratliff et al., 2007) which computes a gradient-like direction. The subgradient of equation (13) is:

$$\frac{\partial J}{\partial \theta} = \frac{1}{m}\sum_{i}(\frac{\partial s(x_i, \hat{y}_{max}, \theta)}{\partial \theta} - \frac{\partial s(x_i, y_i, \theta)}{\partial \theta}) + \lambda\theta$$

where $\hat{y}_{max}$ is the tag sequence with the highest score in equation (13). Following Socher et al. (2013a), we use the diagonal variant of AdaGrad (Duchi et al., 2011) with minibatchs to minimize the objective. The parameter update for the i-th parameter $\theta_{t,i}$ at time step t is as follows:

$$\theta_{t,i} = \theta_{t-1,i} - \frac{\alpha}{\sqrt{\sum_{\tau=1}^{t} g_{\tau,i}^2}} g_{t,i} \qquad (14)$$

where α is the initial learning rate and $g_\tau \in \mathbb{R}^{|\theta_i|}$ is the subgradient at time step τ for parameter θ_i.

	PKU	**MSRA**
Identical words	5.5×10^4	8.8×10^4
Total words	1.1×10^6	2.4×10^6
Identical characters	5×10^3	5×10^3
Total characters	1.8×10^6	4.1×10^6

Table 1: Details of the PKU and MSRA datasets

Window size	$w = 5$
Character(tag) embedding size	$d = 25$
Hidden unit number	$H_2 = 50$
Number of factors	$r = 10$
Initial learning rate	$\alpha = 0.2$
Margin loss discount	$\kappa = 0.2$
Regularization	$\lambda = 10^{-4}$

Table 2: Hyperparameters of our model

4 Experiment

4.1 Data and Model Selection

We use the PKU and MSRA data provided by the second International Chinese Word Segmentation Bakeoff (Emerson, 2005) to test our model. They are commonly used by previous state-of-the-art models and neural network models. Details of the data are listed in Table 1. For evaluation, we use the standard bake-off scoring program to calculate precision, recall, F1-score and out-of-vocabulary (OOV) word recall.

For model selection, we use the first 90% sentences in the training data for training and the rest 10% sentences as development data. The minibatch size is set to 20. Generally, the number of hidden units has a limited impact on the performance as long as it is large enough. We found that 50 is a good trade-off between speed and model performance. The dimensionality of character (tag) embedding is set to 25 which achieved the best performance and faster than 50- or 100-dimensional ones. We also validated on the number of factors for tensor factorization. The performance is not boosted and the training time in-

	P	R	F	OOV
CRF	87.8	85.7	86.7	57.1
NN	92.4	92.2	92.3	60.0
NN+Tag Embed	93.0	92.7	92.9	61.0
MMTNN	**93.7**	**93.4**	**93.5**	**64.2**

Table 3: Test results with different configurations. NN stands for the conventional neural network. NN+Tag Embed stands for the neural network with tag embeddings.

creases drastically when the number of factors is larger than 10. We hypothesize that larger factor size results in too many parameters to train and hence perform worse. The final hyperparameters of our model are set as in Table 2.

4.2 Experiment Results

We first perform a close test[3] on the PKU dataset to show the effect of different model configurations. We also compare our model with the CRF model (Lafferty et al., 2001), which is a widely used log-linear model for Chinese word segmentation. The input feature to the CRF model is simply the context characters (unigram feature) without any additional feature engineering. We use an open source toolkit *CRF++*[4] to train the CRF model. All the neural networks are trained using the Max-Margin approach described in Section 3.4. Table 3 summarizes the test results. As we can see, by using Tag embedding, the F-score is improved by +0.6% and OOV recall is improved by +1.0%, which shows that tag embeddings succeed in modeling the tag-tag interaction and tag-character interaction. Model performance is further boosted after using tensor-based transformation. The F-score is improved by +0.6% while OOV recall is improved by +3.2%, which denotes that tensor-based transformation captures more interactional information than simple non-linear transformation.

Another important result in Table 3 is that our neural network models perform much better than CRF-based model when only unigram features are used. Compared with CRF, there are two differences in neural network models. First, the discrete feature vector is replaced with dense character embeddings. Second, the non-linear transformation is used to discover higher level representation. In fact, CRF can be regarded as a special neural network without non-linear function (Wang and Manning, 2013). Wang and Manning (2013) conduct an empirical study on the effect of non-linearity and the results suggest that non-linear models are highly effective only when distributed representation is used. To explain why distributed representation captures more information than discrete features, we show in Table 4 the effect of character embeddings which are obtained from the lookup table of MMTNN after training. The first row lists three characters we are interested in. In each column, we list the top 5 characters that are nearest (measured by Euclidean distance) to the corresponding character in the first row according to their embeddings. As we can see, characters in the first column are all Chinese number characters and characters in the second column and the third column are all Chinese family names and Chinese punctuations respectively. Therefore, compared with discrete feature representations, distributed representation can capture the syntactic and semantic similarity between characters. As a result, the model can still perform well even if some words do not appear in the training cases.

一(one)	**李(Li)**	**。(period)**
二(two)	赵(Zhao)	，(comma)
三(three)	蒋(Jiang)	：(colon)
四(four)	孔(Kong)	？(question mark)
五(five)	冯(Feng)	“(quotation mark)
六(six)	吴(Wu)	、(Chinese comma)

Table 4: Examples of character embeddings

We further compare our model with previous neural network models on both PKU and MSRA datasets. Since Zheng et al. (2013) did not report the results on the these datasets, we re-implemented their model and tested it on the test data. The results are listed in the first three rows of Table 5, which shows that our model achieved higher F-score than the previous neural network models.

4.3 Unsupervised Pre-training

Previous work found that the performance can be improved by pre-training the character embeddings on large unlabeled data and using the obtained embeddings to initialize the character lookup table instead of random initialization

[3] No other material or knowledge except the training data is allowed

[4] http://crfpp.googlecode.com/svn/trunk/doc/index.html?source=navbar

Models	PKU				MSRA			
	P	**R**	**F**	**OOV**	**P**	**R**	**F**	**OOV**
(Mansur et al., 2013)	87.1	87.9	87.5	48.9	92.3	92.2	92.2	53.7
(Zheng et al., 2013)	92.8	92.0	92.4	63.3	92.9	93.6	93.3	55.7
MMTNN	**93.7**	**93.4**	**93.5**	**64.2**	**94.6**	**94.2**	**94.4**	**61.4**
(Mansur et al., 2013) + Pre-training	91.2	92.7	92.0	68.8	93.1	93.1	93.1	59.7
(Zheng et al., 2013) + Pre-training	93.5	92.2	92.8	69.0	94.2	93.7	93.9	64.1
MMTNN + Pre-training	**94.4**	**93.6**	**94.0**	**69.0**	**95.2**	**94.6**	**94.9**	**64.8**

Table 5: Comparison with previous neural network models

(Mansur et al., 2013; Zheng et al., 2013). Mikolov et al. (2013b) show that pre-trained embeddings can capture interesting semantic and syntactic information such as $king-man+woman \approx queen$ on English data. There are several ways to learn the embeddings on unlabeled data. Mansur et al. (2013) used the model proposed by Bengio et al. (2003) which learns the embeddings based on neural language model. Zheng et al. (2013) followed the model proposed by Collobert et al. (2008). They constructed a neural network that outputs high scores for windows that occur in the corpus and low scores for windows where one character is replaced by a random one. Mikolov et al. (2013a) proposed a faster skip-gram model *word2vec*[5] which tries to maximize classification of a word based on another word in the same sentence. In this paper, we use *word2vec* because preliminary experiments did not show differences between performances of these models but *word2vec* is much faster to train. We pre-train the embeddings on the Chinese Giga-word corpus (Graff and Chen, 2005). As shown in Table 5 (last three rows), both the F-score and OOV recall of our model boost by using pre-training. Our model still outperforms other models after pre-training.

4.4 Minimal Feature Engineering

Although we focus on the question that how far we can go without using feature engineering in this paper, the study of deep learning for NLP tasks is still a new area in which it is currently challenging to surpass the state-of-the-art without additional features. To incorporate features into the neural network, Mansur et al. (2013) proposed the feature-based neural network where each context feature is represented as feature embeddings. The idea of feature embeddings is similar to that of character embeddings described in section 2.1.

[5] https://code.google.com/p/word2vec/

Model	PKU	MSRA
Best05(Chen et al., 2005)	95.0	96.0
Best05(Tseng et al., 2005)	95.0	96.4
(Zhang et al., 2006)	95.1	97.1
(Zhang and Clark, 2007)	94.5	97.2
(Sun et al., 2009)	95.2	97.3
(Sun et al., 2012)	95.4	97.4
(Zhang et al., 2013)	96.1	97.4
MMTNN	94.0	94.9
MMTNN + bigram	95.2	97.2

Table 6: Comparison with state-of-the-art systems

Formally, we assume the extracted features form a feature dictionary D_f. Then each feature $f \in D_f$ is represented by a d-dimensional vector which is called feature embedding. Following their idea, we try to find out how well our model can perform with minimal feature engineering.

A very common feature in Chinese word segmentation is the character bigram feature. Formally, at the i-th character of a sentence $c_{[1:n]}$, the bigram features are $c_k c_{k+1}(i-3 < k < i+2)$. In our model, the bigram features are extracted in the window context and then the corresponding bigram embeddings are concatenated with character embeddings in Layer 1 and fed into Layer 2. In Mansur et al. (2013), the bigram embeddings are pre-trained on unlabeled data with character embeddings, which significantly improves the model performance. Given the long time for pre-training bigram embeddings, we only pre-train the character embeddings and the bigram embeddings are initialized as the average of character embeddings of c_k and c_{k+1}. Further improvement could be obtained if the bigram embeddings are also pre-trained. Table 6 lists the segmentation performances of our model as well as previous state-of-the-art systems. When bigram features are added, the F-score of our model improves

from 94.0% to 95.2% on PKU dataset and from 94.9% to 97.2% on MSRA dataset. It is a competitive result given that our model only use simple bigram features while other models use more complex features. For example, Sun et al. (2012) uses additional word-based features. Zhang et al. (2013) uses eight types of features such as Mutual Information and Accessor Variety and they extract dynamic statistical features from both an in-domain corpus and an out-of-domain corpus using co-training. Since feature engineering is not the main focus of this paper, we did not experiment with more features.

5 Related Work

Chinese word segmentation has been studied with considerable efforts in the NLP community. The most popular approach treats word segmentation as a sequence labeling problem which was first proposed in Xue (2003). Most previous systems address this task by using linear statistical models with carefully designed features such as bigram features, punctuation information (Li and Sun, 2009) and statistical information (Sun and Xu, 2011). Recently, researchers have tended to explore new approaches for word segmentation which circumvent the feature engineering by automatically learning features with neural network models (Mansur et al., 2013; Zheng et al., 2013). Our study is consistent with this line of research, however, our model explicitly models the interactions between tags and context characters and accordingly captures more semantic information.

Tensor-based transformation was also used in other neural network models for its ability to capture multiple interactions in data. For example, Socher et al. (2013b) exploited tensor-based function in the task of Sentiment Analysis to capture more semantic information from constituents. However, given the small size of their tensor matrix, they do not have the problem of high time cost and overfitting problem as we faced in modeling a sequence labeling task like Chinese word segmentation. That's why we propose to decrease computational cost and avoid overfitting with tensor factorization.

Various tensor factorization (decomposition) methods have been proposed recently for tensor-based dimension reduction (Cohen et al., 2013; Van de Cruys et al., 2013; Chang et al., 2013). For example, Chang et al. (2013) proposed the Multi-Relational Latent Semantic Analysis. Similar to LSA, a low rank approximation of the tensor is derived using a tensor decomposition approch. Similar ideas were also used for collaborative filtering (Salakhutdinov et al., 2007) and object recognition (Ranzato et al., 2010). Our tensor factorization is related to these work but uses a different tensor factorization approach. By introducing tensor factorization into the neural network model for sequence labeling tasks, the model training and inference are speeded up and overfitting is prevented.

6 Conclusion

In this paper, we propose a new model called Max-Margin Tensor Neural Network that explicitly models the interactions between tags and context characters. Moreover, we propose a tensor factorization approach that effectively improves the model efficiency and avoids the risk of overfitting. Experiments on the benchmark datasets show that our model achieve better results than previous neural network models and that our model can achieve a competitive result with minimal feature engineering. In the future, we plan to further extend our model and apply it to other structure prediction problems.

Acknowledgments

This work is supported by National Natural Science Foundation of China under Grant No. 61273318 and National Key Basic Research Program of China 2014CB340504.

References

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

Kai-Wei Chang, Wen-tau Yih, and Christopher Meek. 2013. Multi-relational latent semantic analysis. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1602–1612, Seattle, Washington, USA, October. Association for Computational Linguistics.

Aitao Chen, Yiping Zhou, Anne Zhang, and Gordon Sun. 2005. Unigram language model for chinese word segmentation. In *Proceedings of the 4th SIGHAN Workshop on Chinese Language Processing*, pages 138–141. Association for Computational Linguistics Jeju Island, Korea.

Shay B Cohen, Giorgio Satta, and Michael Collins. 2013. Approximate pcfg parsing using tensor decomposition. In *Proceedings of NAACL-HLT*, pages 487–496.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

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

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 999999:2121–2159.

Thomas Emerson. 2005. The second international chinese word segmentation bakeoff. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, volume 133.

David Graff and Ke Chen. 2005. Chinese gigaword. *LDC Catalog No.: LDC2003T09, ISBN*, 1:58563–230.

Geoffrey E Hinton. 1986. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, pages 1–12. Amherst, MA.

Alex Krizhevsky, Geoffrey E Hinton, et al. 2010. Factored 3-way restricted boltzmann machines for modeling natural images. In *International Conference on Artificial Intelligence and Statistics*, pages 621–628.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Zhongguo Li and Maosong Sun. 2009. Punctuation as implicit annotations for chinese word segmentation. *Computational Linguistics*, 35(4):505–512.

Mairgup Mansur, Wenzhe Pei, and Baobao Chang. 2013. Feature-based neural language model and chinese word segmentation. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*.

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

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

Marc'Aurelio Ranzato, Alex Krizhevsky, and Geoffrey E Hinton. 2010. Factored 3-way restricted boltzmann machines for modeling natural images.

Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. 2007. (online) subgradient methods for structured prediction.

Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM.

Holger Schwenk, Anthony Rousseau, and Mohammed Attik. 2012. Large, pruned or continuous space language models on a gpu for statistical machine translation. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 11–19. Association for Computational Linguistics.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013a. Parsing with compositional vector grammars. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

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

Weiwei Sun and Jia Xu. 2011. Enhancing chinese word segmentation using unlabeled data. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 970–979. Association for Computational Linguistics.

Xu Sun, Yaozhong Zhang, Takuya Matsuzaki, Yoshimasa Tsuruoka, and Jun'ichi Tsujii. 2009. A discriminative latent variable chinese segmenter with hybrid word/character information. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 56–64. Association for Computational Linguistics.

Xu Sun, Houfeng Wang, and Wenjie Li. 2012. Fast online training with frequency-adaptive learning rates for chinese word segmentation and new word detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 253–262, Jeju Island,

Korea, July. Association for Computational Linguistics.

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

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter for sighan bakeoff 2005. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, volume 171.

Tim Van de Cruys, Thierry Poibeau, and Anna Korhonen. 2013. A tensor-based factorization model of semantic compositionality. In *Proceedings of NAACL-HLT*, pages 1142–1151.

Mengqiu Wang and Christopher D Manning. 2013. Effect of non-linear deep architecture in sequence labeling. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing*, 8(1):29–48.

Yue Zhang and Stephen Clark. 2007. Chinese segmentation with a word-based perceptron algorithm. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 45, page 840.

Ruiqiang Zhang, Genichiro Kikui, and Eiichiro Sumita. 2006. Subword-based tagging by conditional random fields for chinese word segmentation. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 193–196. Association for Computational Linguistics.

Longkai Zhang, Houfeng Wang, Xu Sun, and Mairgup Mansur. 2013. Exploring representations from unlabeled data with co-training for Chinese word segmentation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 311–321, Seattle, Washington, USA, October. Association for Computational Linguistics.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for Chinese word segmentation and POS tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 647–657, Seattle, Washington, USA, October. Association for Computational Linguistics.

An Empirical Study on the Effect of Negation Words on Sentiment

Xiaodan Zhu, Hongyu Guo, Saif Mohammad and Svetlana Kiritchenko
National Research Council Canada
1200 Montreal Road
Ottawa, K1A 0R6, ON, Canada
{Xiaodan.Zhu,Hongyu.Guo,Saif.Mohammad,Svetlana.Kiritchenko}
@nrc-cnrc.gc.ca

Abstract

Negation words, such as *no* and *not*, play a fundamental role in modifying sentiment of textual expressions. We will refer to a negation word as the *negator* and the text span within the scope of the negator as the *argument*. Commonly used heuristics to estimate the sentiment of negated expressions rely simply on the sentiment of argument (and not on the negator or the argument itself). We use a sentiment treebank to show that these existing heuristics are poor estimators of sentiment. We then modify these heuristics to be dependent on the negators and show that this improves prediction. Next, we evaluate a recently proposed composition model (Socher et al., 2013) that relies on both the negator and the argument. This model learns the syntax and semantics of the negator's argument with a recursive neural network. We show that this approach performs better than those mentioned above. In addition, we explicitly incorporate the prior sentiment of the argument and observe that this information can help reduce fitting errors.

1 Introduction

Morante and Sporleder (2012) define negation to be "a grammatical category that allows the changing of the truth value of a proposition". Negation is often expressed through the use of negative signals or negators–words like *isn't* and *never*, and it can significantly affect the sentiment of its scope. Understanding the impact of negation on sentiment is essential in automatic analysis of sentiment. The literature contains interesting research attempting to model and understand the behavior (reviewed in Section 2). For example,

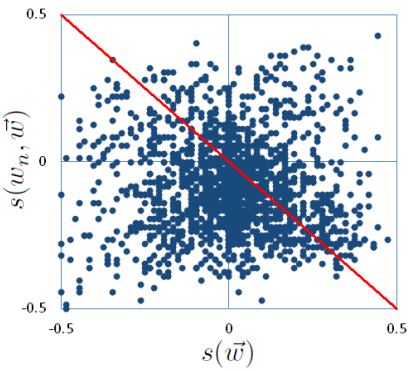


Figure 1: Effect of a list of common negators in modifying sentiment values in Stanford Sentiment Treebank. The x-axis is $s(\vec{w})$, and y-axis is $s(w_n, \vec{w})$. Each dot in the figure corresponds to a text span being modified by (composed with) a negator in the treebank. The red diagonal line corresponds to the sentiment-reversing hypothesis that simply reverses the sign of sentiment values.

a simple yet influential hypothesis posits that a negator reverses the sign of the sentiment value of the modified text (Polanyi and Zaenen, 2004; Kennedy and Inkpen, 2006). The *shifting* hypothesis (Taboada et al., 2011), however, assumes that negators change sentiment values by a constant amount. In this paper, we refer to a negation word as the *negator* (e.g., *isn't*), a text span being modified by and composed with a negator as the *argument* (e.g., *very good*), and entire phrase (e.g., *isn't very good*) as the *negated phrase*.

The recently available Stanford Sentiment Treebank (Socher et al., 2013) renders manually annotated, real-valued sentiment scores for all phrases in parse trees. This corpus provides us with the data to further understand the quantitative behavior of negators, as the effect of negators can now be studied with *arguments* of rich syntactic and semantic variety. Figure 1 illustrates the effect of a common list of negators on sentiment as observed

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 304–313,
Baltimore, Maryland, USA, June 23-25 2014.

on the Stanford Sentiment Treebank.[1] Each dot in the figure corresponds to a *negated phrase* in the treebank. The x-axis is the sentiment score of its *argument* $s(\vec{w})$ and y-axis the sentiment score of the entire negated phrase $s(w_n, \vec{w})$.

We can see that the *reversing* assumption (the red diagonal line) does capture some regularity of human perception, but rather roughly. Moreover, the figure shows that same or similar $s(\vec{w})$ scores (x-axis) can correspond to very different $s(w_n, \vec{w})$ scores (y-axis), which, to some degree, suggests the potentially complicated behavior of negators.[2]

This paper describes a quantitative study of the effect of a list of frequent negators on sentiment. We regard the negators' behavior as an underlying function embedded in annotated data; we aim to model this function from different aspects. By examining sentiment compositions of negators and arguments, we model the quantitative behavior of negators in changing sentiment. That is, given a negated phrase (e.g., *isn't very good*) and the sentiment score of its argument (e.g., $s(\text{“}very\ good'') = 0.5$), we focus on understanding the negator's quantitative behavior in yielding the sentiment score of the negated phrase $s(\text{“}isn't\ very\ good'')$.

We first evaluate the modeling capabilities of two influential heuristics and show that they capture only very limited regularity of negators' effect. We then extend the models to be dependent on the negators and demonstrate that such a simple extension can significantly improve the performance of fitting to the human annotated data. Next, we evaluate a recently proposed composition model (Socher, 2013) that relies on both the negator and the argument. This model learns the syntax and semantics of the negator's argument with a recursive neural network. This approach performs significantly better than those mentioned above. In addition, we explicitly incorporate the prior sentiment of the argument and observe that this information helps reduce fitting errors.

[1]The sentiment values have been linearly rescaled from the original range [0, 1] to [-0.5, 0.5]; in the figure a negative or positive value corresponds to a negative or a positive sentiment respectively; zero means neutral. The negator list will be discussed later in the paper.

[2]Similar distribution is observed in other data such as Tweets (Kiritchenko et al., 2014).

2 Related work

Automatic sentiment analysis The expression of sentiment is an integral component of human language. In written text, sentiment is conveyed with word senses and their composition, and in speech also via prosody such as pitch (Mairesse et al., 2012). Early work on automatic sentiment analysis includes the widely cited work of (Hatzivassiloglou and McKeown, 1997; Pang et al., 2002; Turney, 2002), among others. Since then, there has been an explosion of research addressing various aspects of the problem, including detecting subjectivity, rating and classifying sentiment, labeling sentiment-related semantic roles (e.g., target of sentiment), and visualizing sentiment (see surveys by Pang and Lee (2008) and Liu and Zhang (2012)).

Negation modeling Negation is a general grammatical category pertaining to the changing of the truth values of propositions; negation modeling is not limited to sentiment. For example, paraphrase and contradiction detection systems rely on detecting negated expressions and opposites (Harabagiu et al., 2006). In general, a negated expression and the opposite of the expression may or may not convey the same meaning. For example, *not alive* has the same meaning as *dead*, however, *not tall* does not always mean *short*. Some automatic methods to detect opposites were proposed by Hatzivassiloglou and McKeown (1997) and Mohammad et al. (2013).

Negation modeling for sentiment An early yet influential *reversing* assumption conjectures that a negator reverses the sign of the sentiment value of the modified text (Polanyi and Zaenen, 2004; Kennedy and Inkpen, 2006), e.g., from +0.5 to -0.5, or vice versa. A different hypothesis, called the *shifting* hypothesis in this paper, assumes that negators change the sentiment values by a constant amount (Taboada et al., 2011; Liu and Seneff, 2009). Other approaches to negation modeling have been discussed in (Jia et al., 2009; Wiegand et al., 2010; Lapponi et al., 2012; Benamara et al., 2012).

In the process of semantic composition, the effect of negators could depend on the syntax and semantics of the text spans they modify. The approaches of modeling this include bag-of-word-based models. For example, in the work of (Kennedy and Inkpen, 2006), a feature *not_good* will be created if the word *good* is encountered

within a predefined range after a negator.

There exist different ways of incorporating more complicated syntactic and semantic information. Much recent work considers sentiment analysis from a semantic-composition perspective (Moilanen and Pulman, 2007; Choi and Cardie, 2008; Socher et al., 2012; Socher et al., 2013), which achieved the state-of-the-art performance. Moilanen and Pulman (2007) used a collection of hand-written compositional rules to assign sentiment values to different granularities of text spans. Choi and Cardie (2008) proposed a learning-based framework. The more recent work of (Socher et al., 2012; Socher et al., 2013) proposed models based on recursive neural networks that do not rely on any heuristic rules. Such models work in a bottom-up fashion over the parse tree of a sentence to infer the sentiment label of the sentence as a composition of the sentiment expressed by its constituting parts. The approach leverages a principled method, the forward and backward propagation, to learn a vector representation to optimize the system performance. In principle neural network is able to fit very complicated functions (Mitchell, 1997), and in this paper, we adapt the state-of-the-art approach described in (Socher et al., 2013) to help understand the behavior of negators specifically.

3 Negation models based on heuristics

We begin with previously proposed methods that leverage heuristics to model the behavior of negators. We then propose to extend them to consider lexical information of the negators themselves.

3.1 Non-lexicalized assumptions and modeling

In previous research, some influential, widely adopted assumptions posit the effect of negators to be independent of both the specific negators and the semantics and syntax of the arguments. In this paper, we call a model based on such assumptions a non-lexicalized model. In general, we can simply define this category of models in Equation 1. That is, the model parameters are only based on the sentiment value of the arguments.

$$s(w_n, \vec{w}) \stackrel{\text{def}}{=} f(s(\vec{w})) \quad (1)$$

3.1.1 Reversing hypothesis

A typical model falling into this category is the *reversing* hypothesis discussed in Section 2, where a negator simply reverses the sentiment score $s(\vec{w})$ to be $-s(\vec{w})$; i.e., $f(s(\vec{w})) = -s(\vec{w})$.

3.1.2 Shifting hypothesis

Basic shifting Similarly, a *shifting* based model depends on $s(\vec{w})$ only, which can be written as:

$$f(s(\vec{w})) = s(\vec{w}) - sign(s(\vec{w})) * C \quad (2)$$

where $sign(.)$ is the standard *sign* function which determines if the constant C should be added to or deducted from $s(w_n)$: the constant is added to a negative $s(\vec{w})$ but deducted from a positive one.

Polarity-based shifting As will be shown in our experiments, negators can have different shifting power when modifying a positive or a negative phrase. Thus, we explore the use of two different constants for these two situations, i.e., $f(s(\vec{w})) = s(\vec{w}) - sign(s(\vec{w})) * C(sign(s(\vec{w})))$. The constant C now can take one of two possible values. We will show that this simple modification improves the fitting performance statistically significantly. Note also that instead of determining these constants by human intuition, we use the training data to find the constants in all shifting-based models as well as for the parameters in other models.

3.2 Simple lexicalized assumptions

The above negation hypotheses rely on $s(\vec{w})$. As intuitively shown in Figure 1, the capability of the non-lexicalized heuristics might be limited. Further semantic or syntactic information from either the negators or the phrases they modify could be helpful. The most straightforward way of expanding the non-lexicalized heuristics is probably to make the models to be dependent on the negators.

$$s(w_n, \vec{w}) \stackrel{\text{def}}{=} f(w_n, s(\vec{w})) \quad (3)$$

Negator-based shifting We can simply extend the basic shifting model above to consider the lexical information of negators: $f(s(\vec{w})) = s(\vec{w}) - sign(s(\vec{w})) * C(w_n)$. That is, each negator has its own C. We call this model *negator-based shifting*. We will show that this model also statistically significantly outperforms the basic shifting without overfitting, although the number of parameters have increased.

Combined shifting We further combine the *negator-based shifting* and *polarity-based shifting* above: $f(s(\vec{w})) = s(\vec{w}) - sign(s(\vec{w})) * C(w_n, sign(s(\vec{w})))$. This shifting model is based on negators and the polarity of the text they modify: constants can be different for each negator-polarity pair. The number of parameters in this model is the multiplication of number of negators by two (the number of sentiment polarities). This model further improves the fitting performance on the test data.

4 Semantics-enriched modeling

Negators can interact with arguments in complex ways. Figure 1 shows the distribution of the effect of negators on sentiment without considering further semantics of the arguments. The question then is that whether and how much incorporating further syntax and semantic information can help better fit or predict the negation effect. Above, we have considered the semantics of the negators. Below, we further make the models to be dependent on the arguments. This can be written as:

$$s(w_n, \vec{w}) \stackrel{\text{def}}{=} f(w_n, s(\vec{w}), r(\vec{w})) \quad (4)$$

In the formula, $r(\vec{w})$ is a certain type of representation for the argument $\vec{w}$ and it models the semantics or/and syntax of the argument. There exist different ways of implementing $r(\vec{w})$. We consider two models in this study: one drops $s(\vec{w})$ in Equation 4 and directly models $f(w_n, r(\vec{w}))$. That is, the non-uniform information shown in Figure 1 is not directly modeled. The other takes into account $s(\vec{w})$ too.

For the former, we adopt the recursive neural tensor network (RNTN) proposed recently by Socher et al. (2013), which has showed to achieve the state-of-the-art performance in sentiment analysis. For the latter, we propose a prior sentiment-enriched tensor network (PSTN) to take into account the prior sentiment of the argument $s(\vec{w})$.

4.1 RNTN: Recursive neural tensor network

A recursive neural tensor network (RNTN) is a specific form of feed-forward neural network based on syntactic (phrasal-structure) parse tree to conduct compositional sentiment analysis. For completeness, we briefly review it here. More details can be found in (Socher et al., 2013).

As shown in the *black* portion of Figure 2, each instance of RNTN corresponds to a binary parse tree of a given sentence. Each node of the parse tree is a fixed-length vector that encodes compositional semantics and syntax, which can be used to predict the sentiment of this node. The vector of a node, say p_2 in Figure 2, is computed from the d-dimensional vectors of its two children, namely a and p_1 ($a, p_1 \in \mathbb{R}^{d\times 1}$), with a non-linear function:

$$p_2 = tanh(\begin{bmatrix} a \\ p_1 \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} a \\ p_1 \end{bmatrix} + W \begin{bmatrix} a \\ p_1 \end{bmatrix}) \quad (5)$$

where, $W \in \mathbb{R}^{d\times(d+d)}$ and $V \in \mathbb{R}^{(d+d)\times(d+d)\times d}$ are the matrix and tensor for the composition function. A major difference of RNTN from the conventional recursive neural network (RRN) (Socher et al., 2012) is the use of the tensor V in order to directly capture the multiplicative interaction of two input vectors, although the matrix W *implicitly* captures the nonlinear interaction between the input vectors. The training of RNTN uses conventional forward-backward propagation.

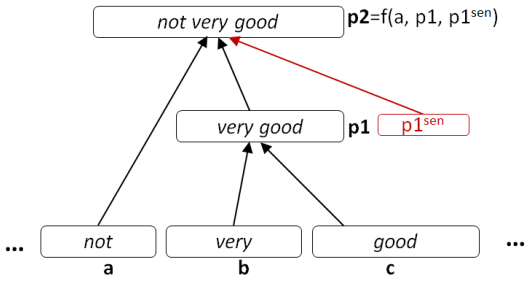


Figure 2: Prior sentiment-enriched tensor network (PSTN) model for sentiment analysis.

4.2 PSTN: Prior sentiment-enriched tensor network

The non-uniform distribution in Figure 1 has showed certain correlations between the sentiment values of $s(w_n, \vec{w})$ and $s(\vec{w})$, and such information has been leveraged in the models discussed in Section 3. We intend to devise a model that implements Equation 4. It bridges between the models we have discussed above that use either $s(\vec{w})$ or $r(\vec{w})$.

We extend RNTN to directly consider the sentiment information of arguments. Consider the node p_2 in Figure 2. When calculating its vector, we aim to directly engage the sentiment information of its right child, i.e., the argument. To this end, we make use of the sentiment class information of

p_1, noted as p_1^{sen}. As a result, the vector of p_2 is calculated as follows:

$$p_2 = tanh(\begin{bmatrix} a \\ p_1 \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} a \\ p_1 \end{bmatrix} + W \begin{bmatrix} a \\ p_1 \end{bmatrix} + \begin{bmatrix} a \\ p_1^{sen} \end{bmatrix}^T V^{sen^{[1:d]}} \begin{bmatrix} a \\ p_1^{sen} \end{bmatrix} + W^{sen} \begin{bmatrix} a \\ p_1^{sen} \end{bmatrix}) \quad (6)$$

As shown in Equation 6, for the node vector $p_1 \in \mathbb{R}^{d\times 1}$, we employ a matrix, namely $W^{sen} \in \mathbb{R}^{d\times(d+m)}$ and a tensor, $V^{sen} \in \mathbb{R}^{(d+m)\times(d+m)\times d}$, aiming at explicitly capturing the interplays between the sentiment class of p_1, denoted as $p_1^{sen}(\in \mathbb{R}^{m\times 1})$, and the negator a. Here, we assume the sentiment task has m classes. Following the idea of Wilson et al. (2005), we regard the sentiment of p_1 as a *prior* sentiment as it has not been affected by the specific context (negators), so we denote our method as prior sentiment-enriched tensor network (PSTN). In Figure 2, the *red* portion shows the added components of PSTN.

Note that depending on different purposes, p_1^{sen} can take the value of the automatically predicted sentiment distribution obtained in forward propagation, the gold sentiment annotation of node p_1, or even other normalized prior sentiment value or confidence score from external sources (e.g., sentiment lexicons or external training data). This is actually an interesting place to extend the current recursive neural network to consider extrinsic knowledge. However, in our current study, we focus on exploring the behavior of negators. As we have discussed above, we will use the human annotated sentiment for the arguments, same as in the models discussed in Section 3.

With the new matrix and tensor, we then have $\theta = (V, V^{sen}, W, W^{sen}, W^{label}, L)$ as the PSTN model's parameters. Here, L denotes the vector representations of the word dictionary.

4.2.1 Inference and Learning

Inference and learning in PSTN follow a forward-backward propagation process similar to that in (Socher et al., 2013), and for completeness, we depict the details as follows. To train the model, one first needs to calculate the predicted sentiment distribution for each node:

$$p_i^{sen} = W^{label} p_i, \quad p_i^{sen} \in \mathbb{R}^{m\times 1}$$

and then compute the posterior probability over the m labels:

$$y^i = \text{softmax}(p_i^{sen})$$

During learning, following the method used by the RNTN model in (Socher et al., 2013), PSTN also aims to minimize the cross-entropy error between the predicted distribution $y^i \in \mathbb{R}^{m\times 1}$ at node i and the target distribution $t^i \in \mathbb{R}^{m\times 1}$ at that node. That is, the error for a sentence is calculated as:

$$E(\theta) = \sum_i \sum_j t_j^i \log y_j^i + \lambda \|\theta\|^2 \quad (7)$$

where, λ represents the regularization hyperparameters, and $j \in m$ denotes the j-th element of the multinomial target distribution.

To minimize $E(\theta)$, the gradient of the objective function with respect to each of the parameters in θ is calculated efficiently via backpropagation through structure, as proposed by Goller and Kchler (1996). Specifically, we first compute the prediction errors in all tree nodes bottom-up. After this forward process, we then calculate the derivatives of the softmax classifiers at each node in the tree in a top-down fashion. We will discuss the gradient computation for the V^{sen} and W^{sen} in detail next. Note that the gradient calculations for the V, W, W^{label}, L are the same as that of presented in (Socher et al., 2013).

In the backpropogation process of the training, each node (except the root node) in the tree carries two kinds of errors: the local softmax error and the error passing down from its parent node. During the derivative computation, the two errors will be summed up as the complement incoming error for the node. We denote the complete incoming error and the softmax error vector for node i as $\delta^{i,com} \in \mathbb{R}^{d\times 1}$ and $\delta^{i,s} \in \mathbb{R}^{d\times 1}$, respectively. With this notation, the error for the root node p_2 can be formulated as follows.

$$\begin{aligned} \delta^{p_2,com} &= \delta^{p_2,s} \\ &= (W^T(y^{p_2} - t^{p_2})) \otimes f'([a; p_1]) \end{aligned} \quad (8)$$

where $\otimes$ is the Hadamard product between the two vectors and f' is the element-wise derivative of $f = tanh$. With the results from Equation 8, we then can calculate the derivatives for the W^{sen} at node p_2 using the following equation:

$$\frac{\partial E^{p_2}}{W^{sen}} = \delta^{p_2,com}([a; p_1^{sen}])^T$$

Similarly, for the derivative of each slice $k(k =$

$1, \ldots, d)$ of the V^{sen} tensor, we have the following:

$$\frac{\partial E^{p_2}}{V^{sen}_{[k]}} = \delta^{p_2,com}_k \begin{bmatrix} a \\ p_1^{sen} \end{bmatrix} \begin{bmatrix} a \\ p_1^{sen} \end{bmatrix}^T$$

Now, let's form the equations for computing the error for the two children of the p_2 node. The difference for the error at p_2 and its two children is that the error for the latter will need to compute the error message passing down from p_2. We denote the error passing down as $\delta^{p_2,down}$, where the left child and the right child of p_2 take the 1^{st} and 2^{nd} half of the error $\delta^{p_2,down}$, namely $\delta^{p_2,down}[1:d]$ and $\delta^{p_2,down}[d+1:2d]$, respectively. Following this notation, we have the error message for the two children of p_2, provided that we have the $\delta^{p_2,down}$:

$$\begin{aligned}\delta^{p_1,com} &= \delta^{p_1,s} + \delta^{p_2,down}[d+1:2d] \\ &= (W^T(y^{p_1} - t^{p_1})) \otimes f^{'}([b;c]) \\ &+ \delta^{p_2,down}[d+1:2d]\end{aligned}$$

The incoming error message of node a can be calculated similarly. Finally, we can finish the above equations with the following formula for computing $\delta^{p_2,down}$:

$$\delta^{p_2,down} = (W^T\delta^{p_2,com}) \otimes f^{'}([a;p_1]) + \delta^{tensor}$$

where

$$\begin{aligned}&\delta^{tensor} = [\delta^V[1:d] + \delta^{V^{sen}}[1:d], \delta^V[d+1:2d]] \\ &= \sum_{k=1}^{d} \delta^{p_2,com}_k (V_{[k]} + (V_{[k]})^T) \otimes f^{'}([a;p_1])[1:d] \\ &+ \sum_{k=1}^{d} \delta^{p_2,com}_k (V^{sen}_{[k]} + (V^{sen}_{[k]})^T) \otimes f^{'}([a;p_1^{sen}])[1:d] \\ &+ \sum_{k=1}^{d} \delta^{p_2,com}_k (V_{[k]} + (V_{[k]})^T) \otimes f^{'}([a;p_1])[d+1:2d]\end{aligned}$$

After the models are trained, they are applied to predict the sentiment of the test data. The original RNTN and the PSTN predict 5-class sentiment for each negated phrase; we map the output to real-valued scores based on the scale that Socher et al. (2013) used to map real-valued sentiment scores to sentiment categories. Specifically, we conduct the mapping with the formula: $p_i^{real} = y^i \cdot [0.1\ 0.3\ 0.5\ 0.7\ 0.9]$; i.e., we calculate the dot product of the posterior probability y^i and the scaling vector. For example, if $y^i = [0.5\ 0.5\ 0\ 0\ 0]$, meaning this phrase has a 0.5 probability to be in the first category (strong negative) and 0.5 for the second category (weak negative), the resulting p_i^{real} will be 0.2 (0.5*0.1+0.5*0.3).

5 Experiment set-up

Data As described earlier, the Stanford Sentiment Treebank (Socher et al., 2013) has manually annotated, real-valued sentiment values for all phrases in parse trees. This provides us with the training and evaluation data to study the effect of negators with syntax and semantics of different complexity in a natural setting. The data contain around 11,800 sentences from movie reviews that were originally collected by Pang and Lee (2005). The sentences were parsed with the Stanford parser (Klein and Manning, 2003). The phrases at all tree nodes were manually annotated with one of 25 sentiment values that uniformly span between the positive and negative poles. The values are normalized to the range of [0, 1].

In this paper, we use a list of most frequent negators that include the words *not*, *no*, *never*, and their combinations with auxiliaries (e.g., *didn't*). We search these negators in the Stanford Sentiment Treebank and normalize the same negators to a single form; e.g., "*is n't*", "*isn't*", and "*is not*" are all normalized to "is_not". Each occurrence of a negator and the phrase it is directly composed with in the treebank, i.e., $\langle w_n, \vec{w} \rangle$, is considered a data point in our study. In total, we collected 2,261 pairs, including 1,845 training and 416 test cases. The split of training and test data is same as specified in (Socher et al., 2013).

Evaluation metrics We use the mean absolute error (MAE) to evaluate the models, which measures the averaged absolute offsets between the predicted sentiment values and the gold standard. More specifically, MAE is calculated as: $MAE = \frac{1}{N}\sum_{\langle w_n,\vec{w}\rangle} |(\hat{s}(w_n,\vec{w}) - s(w_n,\vec{w}))|$, where $\hat{s}(w_n,\vec{w})$ denotes the gold sentiment value and $s(w_n,\vec{w})$ the predicted one for the pair $\langle w_n,\vec{w}\rangle$, and N is the total number of test instances. Note that mean square error (MSE) is another widely used measure for regression, but it is less intuitive for out task here.

6 Experimental results

Overall regression performance Table 1 shows the overall fitting performance of all models. The first row of the table is a random baseline, which

simply guesses the sentiment value for each test case randomly in the range [0,1]. The table shows that the basic *reversing* and *shifting* heuristics do capture negators' behavior to some degree, as their MAE scores are higher than that of the baseline. Making the basic shifting model to be dependent on the negators (model 4) reduces the prediction error significantly as compared with the error of the basic shifting (model 3). The same is true for the polarity-based shifting (model 5), reflecting that the roles of negators are different when modifying positive and negative phrases. Merging these two models yields additional improvement (model 6).

Assumptions	MAE
Baseline	
(1) Random	0.2796
Non-lexicalized	
(2) Reversing	0.1480*
(3) Basic shifting	0.1452*
Simple-lexicalized	
(4) Negator-based shifting	0.1415†
(5) Polarity-based shifting	0.1417†
(6) Combined shifting	0.1387†
Semantics-enriched	
(7) RNTN	0.1097**
(8) PSTN	0.1062††

Table 1: Mean absolute errors (MAE) of fitting different models to Stanford Sentiment Treebank. Models marked with an asterisk (*) are statistically significantly better than the random baseline. Models with a dagger sign (†) significantly outperform model (3). Double asterisks ** indicates a statistically significantly different from model (6), and the model with the double dagger ††is significantly better than model (7). One-tailed paired t-test with a 95% significance level is used here.

Furthermore, modeling the syntax and semantics with the state-of-the-art recursive neural network (model 7 and 8) can dramatically improve the performance over model 6. The PSTN model, which takes into account the human-annotated *prior* sentiment of arguments, performs the best. This could suggest that additional external knowledge, e.g., that from human-built resources or automatically learned from other data (e.g., as in (Kiritchenko et al., 2014)), including sentiment that cannot be inferred from its constituent expressions, might be incorporated to benefit the current neural-network-based models as *prior* knowledge.

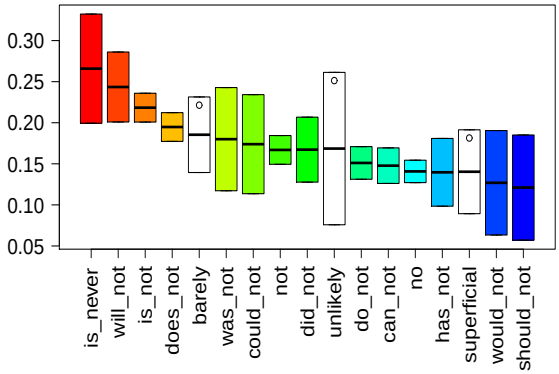


Figure 3: Effect of different negators in shifting sentiment values.

Note that the two neural network based models incorporate the syntax and semantics by representing each node with a vector. One may consider that a straightforward way of considering the semantics of the modified phrases is simply memorizing them. For example, if a phrase *very good* modified by a negator *not* appears in the training and test data, the system can simply memorize the sentiment score of *not very good* in training and use this score at testing. When incorporating this memorizing strategy into model (6), we observed a MAE score of 0.1222. It's not surprising that memorizing the phrases has some benefit, but such matching relies on the exact reoccurrences of phrases. Note that this is a special case of what the neural network based models can model.

Discriminating negators The results in Table 1 has demonstrated the benefit of discriminating negators. To understand this further, we plot in Figure 3 the behavior of different negators: the x-axis is a subset of our negators and the y-axis denotes absolute shifting in sentiment values. For example, we can see that the negator "is_never" on average shifts the sentiment of the arguments by 0.26, which is a significant change considering the range of sentiment value is [0, 1]. For each negator, a 95% confidence interval is shown by the boxes in the figure, which is calculated with the bootstrapping resampling method. We can observe statistically significant differences of shifting abilities between many negator pairs such as that between "*is_never*" and "*do_not*" as well as between "*does_not*" and "*can_not*".

Figure 3 also includes three diminishers (the

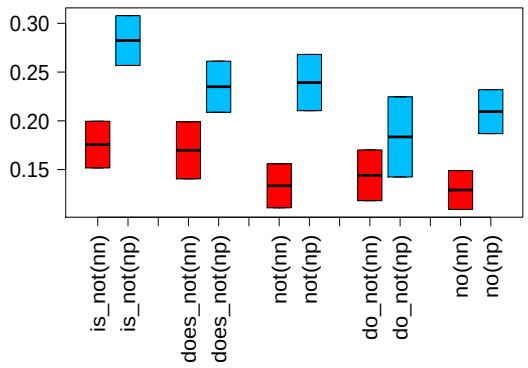


Figure 4: The behavior of individual negators in negated negative (nn) and negated positive (np) context.

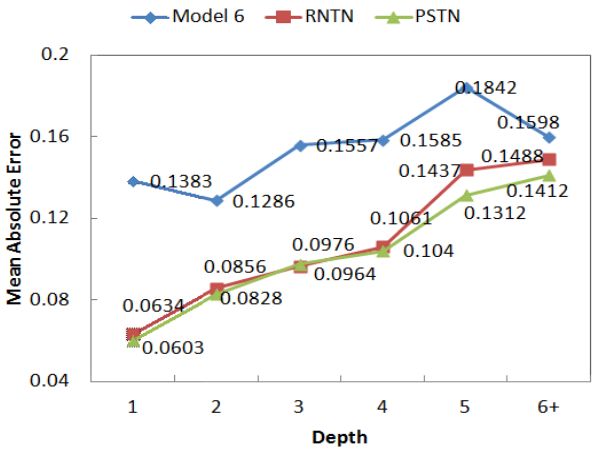


Figure 5: Errors made at different depths in the sentiment tree bank.

white bars), i.e., *barely*, *unlikely*, and *superficial*. By following (Kennedy and Inkpen, 2006), we extracted 319 diminishers (also called *understatement* or *downtoners*) from *General Inquirer*[3]. We calculated their shifting power in the same manner as for the negators and found three diminishers having shifting capability in the shifting range of these negators. This shows that the boundary between negators and diminishers can by fuzzy. In general, we argue that one should always consider modeling negators individually in a sentiment analysis system. Alternatively, if the modeling has to be done in groups, one should consider clustering valence shifters by their shifting abilities in training or external data.

Figure 4 shows the shifting capacity of negators when they modify positive (blue boxes) or negative phrases (red boxes). The figure includes five most frequently used negators found in the sentiment treebank. Four of them have significantly different shifting power when composed with positive or negative phrases, which can explain why the polarity-based shifting model achieves improvement over the basic shifting model.

Modeling syntax and semantics We have seen above that modeling syntax and semantics through the-state-of-the-art neural networks help improve the fitting performance. Below, we take a closer look at the fitting errors made at different depths of the sentiment treebank. The *depth* here is defined as the longest distance between the root of a negator-phrase pair $\langle w_n, \vec{w} \rangle$ and their descendant leafs. Negators appearing at deeper levels of the tree tend to have more complicated syntax and semantics. In Figure 5, the x-axis corresponds to different depths and y-axis is the mean absolute errors (MAE).

The figure shows that both RNTN and PSTN perform much better at all depths than the model 6 in Table 1. When the depths are within 4, the RNTN performs very well and the (human annotated) *prior* sentiment of arguments used in PSTN does not bring additional improvement over RNTN. PSTN outperforms RNTN at greater depths, where the syntax and semantics are more complicated and harder to model. The errors made by model 6 is bumpy, as the model considers no semantics and hence its errors are not dependent on the depths. On the other hand, the errors of RNTN and PSTN monotonically increase with depths, indicating the increase in the task difficulty.

7 Conclusions

Negation plays a fundamental role in modifying sentiment. In the process of semantic composition, the impact of negators is complicated by the syntax and semantics of the text spans they modify. This paper provides a comprehensive and quantitative study of the behavior of negators through a unified view of fitting human annotation. We first measure the modeling capabilities of two influential heuristics on a sentiment treebank and find that they capture some effect of negation; however, extending these non-lexicalized models to be dependent on the negators improves the per-

[3]http://www.wjh.harvard.edu/ inquirer/

formance statistically significantly. The detailed analysis reveals the differences in the behavior among negators, and we argue that they should always be modeled separately. We further make the models to be dependent on the text being modified by negators, through adaptation of a state-of-the-art recursive neural network to incorporate the syntax and semantics of the arguments; we discover this further reduces fitting errors.

References

Farah Benamara, Baptiste Chardon, Yannick Mathieu, Vladimir Popescu, and Nicholas Asher. 2012. How do negation and modality impact on opinions? In *Proceedings of the ACL-2012 Workshop on Extra-Propositional Aspects of Meaning in Computational Linguistics*, pages 10–18, Jeju, Republic of Korea.

Yejin Choi and Claire Cardie. 2008. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 793–801, Honolulu, Hawaii.

Christoph Goller and Andreas Kchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *In Proc. of the ICNN-96*, pages 347–352, Bochum, Germany. IEEE.

Sanda Harabagiu, Andrew Hickl, and Finley Lacatusu. 2006. Negation, contrast and contradiction in text processing. In *AAAI*, volume 6, pages 755–762.

Vasileios Hatzivassiloglou and Kathleen R. McKeown. 1997. Predicting the semantic orientation of adjectives. In *Proceedings of the 8th Conference of European Chapter of the Association for Computational Linguistics*, EACL '97, pages 174–181, Madrid, Spain.

Lifeng Jia, Clement Yu, and Weiyi Meng. 2009. The effect of negation on sentiment analysis and retrieval effectiveness. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 1827–1830, Hong Kong, China. ACM.

Alistair Kennedy and Diana Inkpen. 2006. Sentiment classification of movie reviews using contextual valence shifters. *Computational Intelligence*, 22(2):110–125.

Svetlana Kiritchenko, Xiaodan Zhu, and Saif Mohammad. 2014. Sentiment analysis of short informal texts. *(to appear) Journal of Artificial Intelligence Research*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Sapporo, Japan. Association for Computational Linguistics.

Emanuele Lapponi, Jonathon Read, and Lilja Ovrelid. 2012. Representing and resolving negation for sentiment analysis. In Jilles Vreeken, Charles Ling, Mohammed Javeed Zaki, Arno Siebes, Jeffrey Xu Yu, Bart Goethals, Geoffrey I. Webb, and Xindong Wu, editors, *ICDM Workshops*, pages 687–692. IEEE Computer Society.

Jingjing Liu and Stephanie Seneff. 2009. Review sentiment scoring via a parse-and-paraphrase paradigm. In *EMNLP*, pages 161–169, Singapore.

Bing Liu and Lei Zhang. 2012. A survey of opinion mining and sentiment analysis. In Charu C. Aggarwal and ChengXiang Zhai, editors, *Mining Text Data*, pages 415–463. Springer US.

François Mairesse, Joseph Polifroni, and Giuseppe Di Fabbrizio. 2012. Can prosody inform sentiment analysis? experiments on short spoken reviews. In *ICASSP*, pages 5093–5096, Kyoto, Japan.

Tom M Mitchell. 1997. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 45.

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

Karo Moilanen and Stephen Pulman. 2007. Sentiment composition. In *Proceedings of RANLP 2007*, Borovets, Bulgaria.

Roser Morante and Caroline Sporleder. 2012. Modality and negation: An introduction to the special issue. *Computational linguistics*, 38(2):223–260.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, ACL '05, pages 115–124.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1–2):1–135.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79–86, Philadelphia, USA.

Livia Polanyi and Annie Zaenen. 2004. Contextual valence shifters. In *Exploring Attitude and Affect in Text: Theories and Applications (AAAI Spring Symposium Series)*.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In

Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP '12, Jeju, Korea. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '13, Seattle, USA. Association for Computational Linguistics.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational Linguistics*, 37(2):267–307.

Peter Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *ACL*, pages 417–424, Philadelphia, USA.

Michael Wiegand, Alexandra Balahur, Benjamin Roth, Dietrich Klakow, and Andrés Montoyo. 2010. A survey on the role of negation in sentiment analysis. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing*, NeSp-NLP '10, pages 60–68, Stroudsburg, PA, USA. Association for Computational Linguistics.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 347–354, Stroudsburg, PA, USA. Association for Computational Linguistics.

Extracting Opinion Targets and Opinion Words from Online Reviews with Graph Co-ranking

Kang Liu, Liheng Xu and Jun Zhao
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China
{kliu, lhxu, jzhao}@nlpr.ia.ac.cn

Abstract

Extracting opinion targets and opinion words from online reviews are two fundamental tasks in opinion mining. This paper proposes a novel approach to collectively extract them with graph co-ranking. First, compared to previous methods which solely employed opinion relations among words, our method constructs a heterogeneous graph to model two types of relations, including semantic relations and opinion relations. Next, a co-ranking algorithm is proposed to estimate the confidence of each candidate, and the candidates with higher confidence will be extracted as opinion targets/words. In this way, different relations make cooperative effects on candidates' confidence estimation. Moreover, word preference is captured and incorporated into our co-ranking algorithm. In this way, our co-ranking is personalized and each candidate's confidence is only determined by its preferred collocations. It helps to improve the extraction precision. The experimental results on three data sets with different sizes and languages show that our approach achieves better performance than state-of-the-art methods.

1 Introduction

In opinion mining, extracting opinion targets and opinion words are two fundamental subtasks. Opinion targets are objects about which users' opinions are expressed, and opinion words are words which indicate opinions' polarities. Extracting them can provide essential information for obtaining fine-grained analysis on customers' opinions. Thus, it has attracted a lot of attentions (Hu and Liu, 2004b; Liu et al., 2012; Moghaddam and Ester, 2011; Mukherjee and Liu, 2012).

To this end, previous work usually employed a collective extraction strategy (Qiu et al., 2009; Hu and Liu, 2004b; Liu et al., 2013b). Their intuition is: opinion words usually co-occur with opinion targets in sentences, and there are strong modification relationship between them (called *opinion relation* in (Liu et al., 2012)). If a word is an opinion word, other words with which that word having opinion relations will have highly probability to be opinion targets, and vice versa. In this way, extraction is alternatively performed and mutual reinforced between opinion targets and opinion words. Although this strategy has been widely employed by previous approaches, it still has several limitations.

1) **Only considering opinion relations is insufficient**. Previous methods mainly focused on employing opinion relations among words for opinion target/word co-extraction. They have investigated a series of techniques to enhance opinion relations identification performance, such as nearest neighbor rules (Liu et al., 2005), syntactic patterns (Zhang et al., 2010; Popescu and Etzioni, 2005), word alignment models (Liu et al., 2012; Liu et al., 2013b; Liu et al., 2013a), etc. However, we are curious that whether merely employing opinion relations among words is enough for opinion target/word extraction? We note that there are additional types of relations among words. For example, "*LCD*" and "*LED*" both denote the same aspect "*screen*" in TV set domain, and they are topical related. We call such relations between homogeneous words as semantic relations. If we have known "*LCD*" to be an opinion target, "*LED*" is naturally to be an opinion target. Intuitively, besides opinion relations, semantic relations may provide additional rich clues for indicating opinion targets/words. Which kind of relations is more effective for opinion targets/words extraction? Is it beneficial to consider these two types of relations together for the extraction? To our best knowl-

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 314–324,
Baltimore, Maryland, USA, June 23-25 2014.

edge, these problems have seldom been studied before (see Section 2).

2) **Ignoring word preference**. When employing opinion relations to perform mutual reinforcing extraction between opinion targets and opinion words, previous methods depended on opinion associations among words, but seldom considered *word preference*. Word preference denotes a word's preferred collocations. Intuitively, the confidence of a candidate being an opinion target (opinion word) should mostly be determined by its word preferences rather than all words having opinion relations with it. For example

"*This camera's price is expensive for me.*"

"*It's price is good.*"

"*Canon 40D has a good price.*"

In these three sentences, "*price*" is modified by "*good*" more times than "*expensive*". In traditional extraction strategy, opinion associations are usually computed based on the co-occurrence frequency. Thus, "*good*" has more strong opinion association with "*price*" than "*expensive*", and it would have more contributions on determining "*price*" to be an opinion target or not. It's unreasonable. "*Expensive*" actually has more relatedness with "*price*" than "*good*", and "*expensive*" is likely to be a word preference for "*price*". The confidence of "*price*" being an opinion target should be influenced by "*expensive*" in greater extent than "*good*". In this way, we argue that the extraction will be more precise.

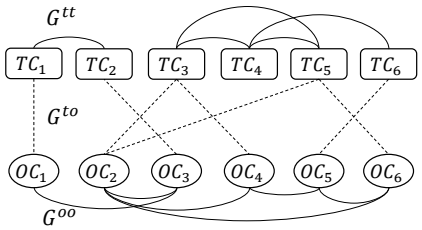


Figure 1: Heterogeneous Graph: OC means opinion word candidates. TC means opinion target candidates. Solid curves and dotted lines respectively mean semantic relations and opinion relations between two candidates.

Thus, to resolve these two problems, we present a novel approach with graph co-ranking. The collective extraction of opinion targets/words is performed in a co-ranking process. First, we operate over a heterogeneous graph to model semantic relations and opinion relations into a unified model. Specifically, our heterogeneous graph is composed of three subgraphs which model different relation types and candidates, as shown in Figure 1. The first subgraph G^{tt} represents semantic relations among opinion target candidates, and the second subgraph G^{oo} models semantic relations among opinion word candidates. The third part is a bipartite subgraph G^{to}, which models opinion relations among different candidate types and connects the above two subgraphs together. Then we perform a random walk algorithm on G^{tt}, G^{oo} and G^{to} separately, to estimate all candidates' confidence, and the entries with higher confidence than a threshold are correspondingly extracted as opinion targets/words. The results could reflect which type of relation is more useful for the extraction.

Second, a co-ranking algorithm, which incorporates three separate random walks on G^{tt}, G^{oo} and G^{to} into a unified process, is proposed to perform candidate confidence estimation. Different relations may cooperatively affect candidate confidence estimation and generate more global ranking results. Moreover, we discover each candidate's preferences through topics. Such word preference will be different for different candidates. We add word preference information into our algorithm and make our co-ranking algorithm be personalized. A candidate's confidence would mainly absorb the contributions from its word preferences rather than its all neighbors with opinion relations, which may be beneficial for improving extraction precision.

We perform experiments on real-world datasets from different languages and different domains. Results show that our approach effectively improves extraction performance compared to the state-of-the-art approaches.

2 Related Work

There are many significant research efforts on opinion targets/words extraction (sentence level and corpus level). In sentence level extraction, previous methods (Wu et al., 2009; Ma and Wan, 2010; Li et al., 2010; Yang and Cardie, 2013) mainly aimed to identify all opinion target/word mentions in sentences. They regarded it as a sequence labeling task, where several classical models were used, such as CRFs (Li et al., 2010) and SVM (Wu et al., 2009).

This paper belongs to corpus level extraction, and aims to generate a sentiment lexicon and a target list rather than to identify mentions in sen-

tences. Most of previous corpus-level methods adopted a co-extraction framework, where opinion targets and opinion words reinforce each other according to their opinion relations. Thus, how to improve opinion relations identification performance was their main focus. (Hu and Liu, 2004a) exploited nearest neighbor rules to mine opinion relations among words. (Popescu and Etzioni, 2005) and (Qiu et al., 2011) designed syntactic patterns to perform this task. (Zhang et al., 2010) promoted Qiu's method. They adopted some special designed patterns to increase recall. (Liu et al., 2012; Liu et al., 2013a; Liu et al., 2013b) employed word alignment model to capture opinion relations rather than syntactic parsing. The experimental results showed that these alignment-based methods are more effective than syntax-based approaches for online informal texts. However, all aforementioned methods only employed opinion relations for the extraction, but ignore considering semantic relations among homogeneous candidates. Moreover, they all ignored word preference in the extraction process.

In terms of considering semantic relations among words, our method is related with several approaches based on topic model (Zhao et al., 2010; Moghaddam and Ester, 2011; Moghaddam and Ester, 2012a; Moghaddam and Ester, 2012b; Mukherjee and Liu, 2012). The main goals of these methods weren't to extract opinion targets/words, but to categorize all given aspect terms and sentiment words. Although these models could be used for our task according to the associations between candidates and topics, solely employing semantic relations is still one-sided and insufficient to obtain expected performance.

Furthermore, there is little work which considered these two types of relations globally (Su et al., 2008; Hai et al., 2012; Bross and Ehrig, 2013). They usually captured different relations using co-occurrence information. That was too coarse to obtain expected results (Liu et al., 2012). In addition, (Hai et al., 2012) extracted opinion targets/words in a bootstrapping process, which had an error propagation problem. In contrast, we perform extraction with a global graph co-ranking process, where error propagation can be effectively alleviated. (Su et al., 2008) used heterogeneous relations to find implicit sentiment associations among words. Their aim was only to perform aspect terms categorization but not to extract opinion targets/words. They extracted opinion targets/words in advanced through simple phrase detection. Thus, the extraction performance is far from expectation.

3 The Proposed Method

In this section, we propose our method in detail. We formulate opinion targets/words extraction as a co-ranking task. All nouns/noun phrases are regarded as opinion target candidates, and all adjectives/verbs are regarded as opinion word candidates, which are widely adopted by pervious methods (Hu and Liu, 2004a; Qiu et al., 2011; Wang and Wang, 2008; Liu et al., 2012). Then each candidate will be assigned a confidence and ranked, and the candidates with higher confidence than a threshold will be extracted as the results.

Different from traditional methods, besides opinion relations among words, we additionally capture semantic relations among homogeneous candidates. To this end, a heterogeneous undirected graph $G = (V, E)$ is constructed. $V = V^t \cup V^o$ denotes the vertex set, which includes opinion target candidates $v^t \in V^t$ and opinion word candidates $v^o \in V^o$. E denotes the edge set, where $e_{ij} \in E$ means that there is a relation between two vertices. $E^{tt} \subset E$ represents the semantic relations between two opinion target candidates. $E^{oo} \subset E$ represents the semantic relations between two opinion word candidates. $E^{to} \subset E$ represents the opinion relations between opinion target candidates and opinion word candidates. Based on different relation types, we used three matrices $M_{tt} \in \mathbb{R}^{|V^t|\times|V^t|}$, $M_{oo} \in \mathbb{R}^{|V^o|\times|V^o|}$ and $M_{to} \in \mathbb{R}^{|V^t|\times|V^o|}$ to record the association weights between any two vertices, respectively. Section 3.4 will illustrate how to construct them.

3.1 Only Considering Opinion Relations

To estimate the confidence of each candidate, we use a random walk algorithm on our graph to perform co-ranking. Most previous methods (Hu and Liu, 2004a; Qiu et al., 2011; Wang and Wang, 2008; Liu et al., 2012) only considered opinion relations among words. Their basic assumption is as follows.

> **Assumption 1:** If a word is likely to be an opinion word, the words which it has opinion relation with will have higher confidence to be opinion targets, and vice versa.

In this way, candidates' confidences (v^t or v^o) are collectively determined by each other iteratively. It equals to making random walk on subgraph $G^{to} = (V, E^{to})$ of G. Thus we have

$$\begin{aligned} C_t &= (1-\mu) \times M_{to} \times C_o + \mu \times I_t \\ C_o &= (1-\mu) \times M_{to}^T \times C_t + \mu \times I_o \end{aligned} \tag{1}$$

where C_t and C_o respectively represent confidences of opinion targets and opinion words. $m_{i,j}^{to} \in M_{to}$ means the association weight between the ith opinion target and the jth opinion word according to their opinion relations.

It's worthy noting that I_t and I_o respectively denote prior confidences of opinion target candidates and opinion word candidates. We argue that opinion targets are usually domain-specific, and there are remarkably distribution difference of them on different domains (in-domain D_{in} vs. out-domain D_{out}). If a candidate is salient in D_{in} but common in D_{out}, it's likely to be an opinion target in D_{in}. Thus, we use a domain relevance measure (DR) (Hai et al., 2013) to compute I_t.

$$DR(t) = \frac{R(t, D_{in})}{R(t, D_{out})} \tag{2}$$

where $R(t, D) = \frac{\bar{w}_t}{s_t} \times \sum_{j=1}^{N}(w_{tj} - \frac{1}{W_j} \times \sum_{k=1}^{W_j} w_{kj})$ represents candidate relevance with domain D. $w_{tj} = (1 + logTF_{tj}) \times log\frac{N}{DF_t}$ is a TF-IDF-like weight of candidate t in document j. TF_{tj} is the frequency of the candidate t in the jth document, and DF_t is document frequency. N means the document number in domain D. $R(t, D)$ includes two measures to reflect the salient of a candidate in D. 1) $w_{tj} - \frac{1}{W_j} \times \sum_{k=1}^{W_j} w_{kj}$ reflects how frequently a term is mentioned in a particular document. W_j denotes the word number in document j. 2) $\frac{\bar{w}_t}{s_t}$ quantifies how significantly a term is mentioned across all documents in D. $\bar{w}_t = \frac{1}{N} \times \sum_{k=1}^{N} w_{tk}$ denotes average weight across all documents for t. $s_t = \sqrt{\frac{1}{N} \times \sum_{j=1}^{N} (w_{tj} - \bar{w}_j)^2}$ denotes the standard variance of term t. We use the given reviews as in-domain collection D_{in} and Google n-gram corpus[1] as out-domain collection D_{out}. Finally, each entry in I_t is a normalized $DR(t)$ score. In contrast, opinion words are usually domain-independent. Users may use same words to express theirs opinions, like "good", "bad", etc. But there are still some domain-dependent opinion words, like "delicious" in the restaurant domain, "powerful" in the car domain. It's difficult to discriminate them from other words by using statistical information. So we simply set all entries in I_o to be 1. $\mu \in [0, 1]$ in Eq.1 determines the impact of the prior confidence on results.

[1] http://books.google.com/ngrams/datasets

3.2 Only Considering Semantic Relations

To estimate candidates' confidences by only considering semantic relations among words, we make two separately random walks on the subgraphs of G, $G^{tt} = (V, E^{tt})$ and $G^{oo} = (V, E^{oo})$. The basic assumption is as follows:

> **Assumption 2:** If a word is likely to be an opinion target (opinion word), the words which it has strong semantic relation with will have higher confidence to be opinion targets (opinion words).

In this way, the confidence of the candidate is determined only by its homogeneous neighbours. There is no mutual reinforcement between opinion targets and opinion words. Thus we have

$$\begin{aligned} C_t &= (1-\nu) \times M_{tt} \times C_t + \nu \times I_t \\ C_o &= (1-\nu) \times M_{oo} \times C_o + \nu \times I_o \end{aligned} \tag{3}$$

where ν has the same role as μ in Eq.1.

3.3 Considering Semantic Relations and Opinion Relations Together

To jointly model semantic relations and opinion relations for opinion targets/words extraction, we couple two random walking algorithms mentioned above together. Here, **Assumption 1** and **Assumption 2** are both satisfied. Thus, an opinion target/word candidate's confidence is collectively determined by its neighbours according to different relation types. Meanwhile, each item may make influence on it's neighbours. It's an iterative reinforcement process. Thus, we have

$$\begin{aligned} C_t &= (1-\lambda-\mu) \times M_{to} \times C_o \\ &\quad + \lambda \times M_{tt} \times C_t + \mu \times I_t \\ C_o &= (1-\lambda-\mu) \times M_{to}^T \times C_t \\ &\quad + \lambda \times M_{oo} \times C_o + \mu \times I_o \end{aligned} \tag{4}$$

where $\lambda \in [0, 1]$ determines which type of relations dominates candidate confidence estimation. $\lambda = 0$ means that each candidate's confidence is estimated by only considering opinion relations among words, which equals to Eq.1. Otherwise, when $\lambda = 1$, candidate confidence estimation only

considers semantic relations among words, which equals to Eq.3. μ, I_o and I_t have the same meaning in Eq.1. Our algorithm will run iteratively until it converges or in a fixed iteration number $Iter$. In experiments, we set $Iter = 200$.

Obtaining Word Preference. The co-ranking algorithm in Eq.4 is based on a standard random walking algorithm, which randomly selects a link according to the association matrix M_{to}, M_{tt} and M_{oo}, or jumps to a random node with prior confidence value. However, it generates a global ranking over all candidates without taking the node preference (word preference) into account. As mentioned in the first section, each opinion target/word has its preferred collocations, it's reasonable that the confidence of an opinion target (opinion word) candidate should be preferentially determined by its preferences, rather than all of its neighbors with opinion relations.

To obtain the word preference, we resort to topics. We believe that if an opinion word ${v_i}^o$ is topical related with a target word ${v_j}^t$, ${v_i}^o$ can be regarded as a word preference for ${v_j}^t$, and vice versa. For example, "*price*" and "*expensive*" are topically related in phone's domain, so they are a word preference for each other.

Specifically, we use a vector $P^{T_i} = [P_1^{T_i}, ..., P_k^{T_i}, ..., P_{|V^o|}^{T_i}]_{1\times|V^o|}$ to represent word preference of the ith opinion target candidate. $P_k^{T_i}$ means the preferred probability of the ith potential opinion target for the kth potential opinion words. To compute $P_k^{T_i}$, we first use *Kullback-Leibler* divergence to measure the semantic distance between any two candidates on the bridge of topics. Thus, we have

$$D(v_i, v_j) = \frac{1}{2}\Sigma_z(KL_z(v_i||v_j) + KL_z(v_j||v_i))$$

where $KL_z(v_i||v_j) = p(z|v_i)log\frac{p(z|v_i)}{p(z|v_j)}$ means the KL-divergence from candidate v_i to v_j based on topic z. $p(z|v) = p(v|z)\frac{p(z)}{p(v)}$, where $p(v|z)$ is the probability of the candidate v to topic z (see Section 3.4). $p(z)$ is the probability that topic z in reviews. $p(v)$ is the probability that a candidate occurs in reviews. Then, a logistic function is used to map $D(v_i, v_j)$ into $[0, 1]$.

$$SA(v_i, v_j) = \frac{1}{1 + e^{D(v_i,v_j)}} \quad (5)$$

Then, we calculate $P_k^{T_i}$ by normalize $SA(v_i, v_j)$ score, i.e. $P_k^{T_i} = \frac{SA(v_i^t, v_k^o)}{\sum_{p=1}^{|V^o|} SA(v_i^t, v_p^o)}$. For demonstration, we give some examples in Table 1, where each entry denotes a $SA(v_i, v_j)$ score between two candidates. We can see that using topics can successfully capture the preference information for each opinion target/word.

	expensive	good	long	colorful
price	0.265	0.043	0.003	0.000
LED	0.002	0.035	0.007	0.098
battery	0.000	0.015	0.159	0.001

Table 1: Examples of Calculated Word Preference

And we use a vector $P^{O_j} = [P_1^{O_j}, ..., P_q^{O_j}, ..., P_{|V^t|}^{O_j}]_{1\times|V^t|}$ to represent the preference information of the jth opinion word candidate. Similarly, we have $P_q^{O_j} = \frac{SA(v_q^t, v_j^o)}{\sum_{k=1}^{|V^t|} SA(v_k^t, v_j^o)}$.

Incorporating Word Preference into Co-ranking. To consider such word preference in our co-ranking algorithm, we incorporate it into the random walking on G^{to}. Intuitively, preference vectors will be different for different candidates. Thus, the co-ranking algorithm would be personalized. It allows that the candidate confidence propagates to other candidates only in its preference cluster. Specifically, we make modification on original transition matrix $M_{to} = (M_1^{to}, M_2^{to}, ..., M_{|V^t|}^{to})$ and add each candidate's preference in it. Let $\hat{M}_{to} = (\hat{M}_1^{to}, \hat{M}_2^{to}, ..., \hat{M}_{|V^t|}^{to})$ be the modified transition matrix, which records the associations between opinion target candidates and opinion word candidates. Here $M_k^{to} \in \mathbb{R}^{1\times|V^o|}$ and $\hat{M}_k^{to} \in \mathbb{R}^{1\times|V^o|}$ denotes the kth column vector in M_{to} and $\hat{M}_{to}$, respectively. And let $Diag(P^{T_k})$ denote a diagonal matrix whose eigenvalue is vector P^{T_k}, we have

$$\hat{M}_k^{to} = M_k^{to} Diag(P^{T_k})$$

Similarly, let $U_k^{to} \in \mathbb{R}^{1\times|V^t|}$ and $\hat{U}_k^{to} \in \mathbb{R}^{1\times|V^t|}$ denotes the kth row vector in M_{to}^T and $\hat{M}_{to}^T$, respectively. $Diag(P^{O_k})$ denote a diagonal matrix whose eigenvalue is vector P^{O_k}. Then we have

$$\hat{U}_k^{to} = U_k^{to} Diag(P^{O_k})$$

In this way, each candidate's preference is incorporated into original associations based on opinion relation M_{to} through $Diag(P^{O_k})$ and $Diag(P^{T_k})$. And candidates' confidences will mainly come from the contributions of its preferences. Thus, C_t and C_o in Eq.4 become:

$$
\begin{aligned}
C_t &= (1-\lambda-\mu) \times \hat{M}_{to} \times C_o \\
&\quad + \lambda \times M_{tt} \times C_t + \mu \times I_t \\
C_o &= (1-\lambda-\mu) \times \hat{M}_{to}^T \times C_t \\
&\quad + \lambda \times M_{oo} \times C_o + \mu \times I_o
\end{aligned} \tag{6}
$$

3.4 Capturing Semantic and Opinion Relations

In this section, we explain how to capture semantic relations and opinion relations for constructing transition matrices M_{tt}, M_{oo} and M_{to}.

Capturing Semantic Relations: For capturing semantic relations among homogenous candidates, we employ topics. We believe that if two candidates share similar topics in the corpus, there is a strong semantic relation between them. Thus, we employ a LDA variation (Mukherjee and Liu, 2012), an extension of (Zhao et al., 2010), to discover topic distribution on words, which sampled all words into two separated observations: opinion targets and opinion words. It's because that we are only interested in topic distribution of opinion targets/words, regardless of other useless words, including conjunctions, prepositions etc. This model has been proven to be better than the standard LDA model and other LDA variations for opinion mining (Mukherjee and Liu, 2012).

After topic modeling, we obtain the probability of the candidates (v^t and v^o) to topic z, i.e. $p(z|v^t)$ and $p(z|v^o)$, and topic distribution $p(z)$. Then, a symmetric *Kullback-Leibler* divergence as same as Eq.5 is used to calculate the semantical associations between any two homogenous candidates. Thus, we obtain $SA(v^t, v^t)$ and $SA(v^o, v^o)$, which correspond to the entries in M_{tt} and M_{oo}, respectively.

Capturing Opinion Relations: To capture opinion relations among words and construct the transition matrix M_{to}, we used an alignment-based method proposed in (Liu et al., 2013b). This approach models capturing opinion relations as a monolingual word alignment process. Each opinion target can find its corresponding modifiers in sentences through alignment, in which multiple factors are considered globally, such as co-occurrence information, word position in sentence, etc. Moreover, this model adopted a partially supervised framework to combine syntactic information with alignment results, which has been proven to be more precise than the state-of-the-art approaches for opinion relations identification (Liu et al., 2013b).

After performing word alignment, we obtain a set of word pairs composed of a noun (noun phrase) and its corresponding modified word. Then, we simply employ Pointwise Mutual Information (PMI) to calculate the opinion associations among words as the entries in M_{to}. $OA(v^t, v^o) = log\frac{p(v^t, v^o)}{p(v^t)p(v^o)}$, where v^t and v^o denote an opinion target candidate and an opinion word candidate, respectively. $p(v^t, v^o)$ is the co-occurrence probability of v^t and v^o based on the opinion relation identification results. $p(v^t)$ and $p(v^o)$ give the independent occurrence probability of of v^t and v^o, respectively

4 Experiments

4.1 Datasets and Evaluation Metrics

Datasets: To evaluate the proposed method, we used three datasets. The first one is *Customer Review Datasets* (CRD), used in (Hu and Liu, 2004a), which contains reviews about five products. The second one is *COAE2008* dataset2[2], which contains Chinese reviews about four products. The third one is *Large*, also used in (Wang et al., 2011; Liu et al., 2012; Liu et al., 2013a), where two domains are selected (Mp3 and Hotel). As mentioned in (Liu et al., 2012), *Large* contains 6,000 sentences for each domain. Opinion targets/words are manually annotated, where three annotators were involved. Two annotators were required to annotate out opinion words/targets in reviews. When conflicts occur, the third annotator make final judgement. In total, we respectively obtain 1,112, 1,241 opinion targets and 334, 407 opinion words in Hotel, MP3.

Pre-processing: All sentences are tagged to obtain words' part-of-speech tags using Stanford NLP tool[3]. And noun phrases are identified using the method in (Zhu et al., 2009) before extraction.

Evaluation Metrics: We select precision(P), recall(R) and f-measure(F) as metrics. And a significant test is performed, i.e., a t-test with a default significant level of 0.05.

4.2 Our Method vs. The State-of-the-art Methods

To prove the effectiveness of the proposed method, we select some state-of-the-art methods for comparison as follows:

[2]http://ir-china.org.cn/coae2008.html

[3]http://nlp.stanford.edu/software/tagger.shtml

Methods	D1			D2			D3			D4			D5			Avg.
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	F
Hu	0.75	0.82	0.78	0.71	0.79	0.75	0.72	0.76	0.74	0.69	0.82	0.75	0.74	0.80	0.77	0.758
DP	**0.87**	0.81	0.84	**0.90**	0.81	0.85	**0.90**	0.86	0.88	**0.81**	0.84	0.82	**0.92**	0.86	0.89	0.856
Zhang	0.83	0.84	0.83	0.86	0.85	0.85	0.86	0.88	0.87	0.80	0.85	0.82	0.86	0.86	0.86	0.846
SAS	0.80	0.79	0.79	0.82	0.76	0.79	0.79	0.74	0.76	0.77	0.78	0.77	0.80	0.76	0.78	0.778
Liu	0.84	0.85	0.84	0.87	0.85	0.86	0.88	0.89	0.88	**0.81**	0.85	**0.83**	0.89	0.87	0.88	0.858
Hai	0.77	**0.87**	0.83	0.79	**0.86**	0.82	0.79	0.89	0.84	0.72	0.88	0.79	0.74	0.88	0.81	0.818
CR	0.84	0.86	0.85	0.87	0.85	0.86	0.87	**0.90**	0.88	**0.81**	0.87	**0.83**	0.89	0.88	0.89	0.862
CR_WP	0.86	0.86	**0.86**	0.88	**0.86**	**0.87**	0.89	**0.90**	**0.89**	**0.81**	0.87	**0.83**	0.91	**0.89**	**0.90**	**0.870**

Table 2: Results of Opinion Targets Extraction on *Customer Review Dataset*

Methods	Camera			Car			Laptop			Phone			Mp3			Hotel			Avg.
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	F
Hu	0.63	0.65	0.64	0.62	0.58	0.60	0.51	0.67	0.58	0.69	0.60	0.64	0.61	0.68	0.64	0.60	0.65	0.62	0.587
DP	0.71	0.70	0.70	0.72	0.65	0.68	0.58	0.69	0.63	0.78	0.66	0.72	0.69	0.70	0.69	0.67	0.69	0.68	0.683
Zhang	0.71	0.78	0.74	0.69	0.68	0.68	0.57	0.80	0.67	0.80	0.71	0.75	0.67	0.77	0.72	0.67	0.76	0.71	0.712
SAS	0.72	0.72	0.72	0.71	0.64	0.67	0.59	0.72	0.65	0.78	0.69	0.73	0.69	0.75	0.72	0.69	0.74	0.71	0.700
Liu	0.75	0.81	0.78	0.71	0.71	0.71	0.61	0.85	0.71	0.83	0.74	0.78	0.70	0.82	0.76	0.71	0.80	0.75	0.749
Hai	0.68	**0.84**	0.76	0.69	**0.75**	0.72	0.58	**0.86**	0.72	0.75	0.76	0.76	0.65	0.83	0.74	0.62	0.82	0.75	0.742
CR	0.75	0.83	0.79	0.72	0.74	0.73	0.60	0.85	0.70	0.83	**0.77**	**0.80**	0.70	**0.84**	0.76	0.71	**0.83**	0.77	0.758
CR_WP	**0.78**	**0.84**	**0.81**	**0.74**	**0.75**	**0.74**	**0.64**	0.85	**0.73**	**0.84**	0.76	**0.80**	**0.74**	**0.84**	**0.79**	**0.74**	0.82	**0.78**	**0.773**

Table 3: Results of Opinion Targets Extraction on *COAE 2008* and *Large*

Hu extracted opinion targets/words using association mining rules (Hu and Liu, 2004a).

DP used syntax-based patterns to capture opinion relations in sentences, and then used a bootstrapping process to extract opinion targets/words (Qiu et al., 2011),.

Zhang is proposed by (Zhang et al., 2010). They also used syntactic patterns to capture opinion relations between words. Then a HITS (Kleinberg, 1999) algorithm is employed to extract opinion targets.

Liu is proposed by (Liu et al., 2013a), an extension of (Liu et al., 2012). They employed a word alignment model to capture opinion relations among words, and then used a random walking algorithm to extract opinion targets.

Hai is proposed by (Hai et al., 2012), which is similar to our method. They employed both of semantic relations and opinion relations to extract opinion words/targets in a bootstrapping framework. But they captured relations only using co-occurrence statistics. Moreover, word preference was not considered.

SAS is proposed by (Mukherjee and Liu, 2012), an extended lda-based model of (Zhao et al., 2010). The top K items for each aspect are extracted as opinion targets/words. It means that only semantic relations among words are considered in SAS. And we set aspects number to be 9 as same as (Mukherjee and Liu, 2012).

CR: is the proposed method in this paper by using co-ranking, referring to Eq.4. CR doesn't consider word preference.

CR_WP: is the full implementation of our method, referring to Eq.6.

Hu, DP, Zhang and Liu are the methods which only consider opinion relations among words. SAS is the methods which only consider semantic relations among words. Hai, CR and CR_WP consider these two types of relations together. The parameter settings of state-of-the-art methods are same as their original paper. In CR and CR_WP, we set $\lambda = 0.4$ and $\mu = 0.1$. The experimental results are shown in Table 2, 3, 4 and 5, where the last column presents the average F-measure scores for multiple domains. Since Liu and Zhang aren't designed for opinion words extraction, we don't present their results in Table 4 and 5. From experimental results, we can see.

1) Our methods (CR and CR_WP) outperform other methods not only on opinion targets extraction but on opinion words extraction in most domains. It proves the effectiveness of the proposed method.

2) CR and CR_WP have much better performance than Liu and Zhang, especially on Recall. Liu and Zhang also use a ranking framework like ours, but they only employ opinion relations for extraction. In contrast, besides opinion relations, CR and CR_WP further take semantic relations into account. Thus, more opinion targets/words can be extracted. Furthermore, we observe that CR and CR_WP outperform SAS. SAS only exploits semantic relations, but ignores opinion relations among words. Its extraction is performed separately and neglects the reinforcement between opinion targets and opinion words. Thus, SAS has worse performance than our methods. It demonstrates the usefulness of considering multiple relation types.

Methods	D1			D2			D3			D4			D5			Avg.
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	F
Hu	0.57	0.75	0.65	0.51	0.76	0.61	0.57	0.73	0.64	0.54	0.62	0.58	0.62	0.67	0.64	0.624
DP	0.64	0.73	0.68	0.57	0.79	0.66	0.65	0.70	0.67	0.61	0.65	0.63	0.70	0.68	0.69	0.666
SAS	0.64	0.68	0.66	0.55	0.70	0.62	0.62	0.65	0.63	0.60	0.61	0.60	0.68	0.63	0.65	0.632
Hai	0.62	**0.77**	0.69	0.52	**0.80**	0.64	0.60	0.74	0.67	0.56	**0.69**	0.62	0.66	**0.70**	0.68	0.660
CR	0.62	0.75	0.68	0.57	0.79	0.67	0.64	**0.75**	0.69	0.63	**0.69**	0.66	0.68	0.69	0.69	0.678
CR_WP	**0.65**	0.75	**0.70**	**0.59**	**0.80**	**0.68**	**0.65**	0.74	**0.70**	**0.66**	0.68	**0.67**	**0.71**	**0.70**	**0.70**	**0.690**

Table 4: Results of Opinion Words Extraction on *Customer Review Dataset*

Methods	Camera			Car			Laptop			Phone			Mp3			Hotel			Avg.
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	F
Hu	0.72	0.74	0.73	0.70	0.71	0.70	0.66	0.70	0.68	0.70	0.70	0.70	0.48	0.67	0.56	0.52	0.69	0.59	0.660
DP	0.80	0.73	0.76	0.79	0.71	0.75	0.75	0.69	0.72	0.78	0.68	0.73	0.60	0.65	0.62	0.61	0.66	0.63	0.702
SAS	0.73	0.70	0.71	0.75	0.68	0.71	0.72	0.68	0.69	0.71	0.66	0.68	0.64	0.62	0.63	0.66	0.61	0.63	0.675
Hai	0.76	0.74	0.75	0.72	**0.74**	0.73	0.69	**0.72**	0.70	0.72	0.70	0.71	0.61	**0.69**	0.64	0.59	0.68	0.64	0.690
CR	**0.80**	**0.75**	**0.77**	0.77	**0.74**	0.75	0.73	0.71	0.72	0.75	0.71	0.73	0.63	**0.69**	0.64	0.63	0.68	0.66	0.710
CR_WP	**0.80**	**0.75**	**0.77**	**0.80**	**0.74**	**0.77**	**0.77**	0.71	**0.74**	**0.78**	**0.72**	**0.75**	**0.66**	0.68	**0.67**	**0.67**	**0.69**	**0.68**	**0.730**

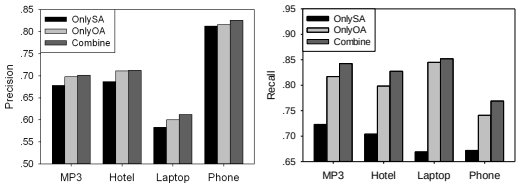
Table 5: Results of Opinion Words Extraction on *COAE 2008* and *Large*

3) CR and CR_WP both outperform Hai. We believe the reasons are as follows. First, CR and CR_WP considers multiple relations in a unified process by using graph co-ranking. In contrast, Hai adopts a bootstrapping framework which performs extraction step by step and may have the problem of error propagation. It demonstrates that our graph co-ranking is more suitable for this task than bootstrapping-based strategy. Second, our method captures semantic relations using topic modeling and captures opinion relations through word alignments, which are more precise than Hai which merely uses co-occurrence information to indicate such relations among words. In addition, word preference is not handled in Hai, but processed in CR_WP. The results show the usefulness of word preference for opinion targets/words extraction.

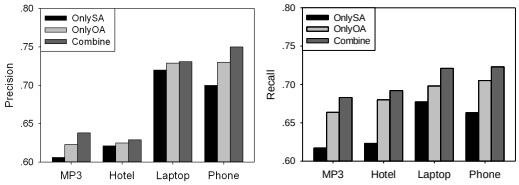
4) CR_WP outperforms CR, especially on precision. The only difference between them is that CR_WP considers word preference when performing graph ranking for candidate confidence estimation, but CR does not. Each candidate confidence estimation in CR_WP gives more weights for this candidate's preferred words than CR. Thus, the precision can be improved.

4.3 Semantic Relation vs. Opinion Relation

In this section, we discuss which relation type is more effective for this task. For comparison, we design two baselines, called **OnlySA** and **OnlyOA**. OnlyOA only employs opinion relations among words, which equals to Eq.1. OnlySA only employs semantic relations among words, which equals to Eq.3. Moreover, **Combine** is our method which considers both of opinion relations and semantic relations together, referring to Eq.4 with



(a) Opinion Target Extraction Results



(b) Opinion Word Extraction Results

Figure 2: Semantic Relations vs. Opinion Relations

$\lambda = 0.5$. Figure 2 presents experimental results. The left graph presents opinion targets extraction results and the right graph presents opinion words extraction results. Because of space limitation, we only shown the results of four domains (MP3, Hotel, Laptop and Phone).

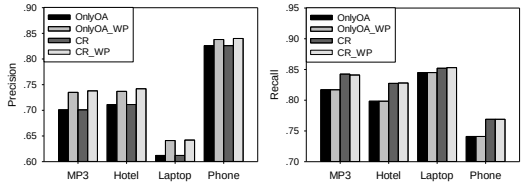
From results, we observe that OnlyOA outperforms OnlySA in all domains. It demonstrates that employing opinion relations are more useful than semantic relations for co-extracting opinion targets/words. And it is necessary to utilize the mutual reinforcement relationship between opinion words and opinion targets. Moreover, Combine outperforms OnlySA and OnlyOA in all domains. It indicates that combining different relations among words together is effective.

4.4 The Effectiveness of Considering Word Preference

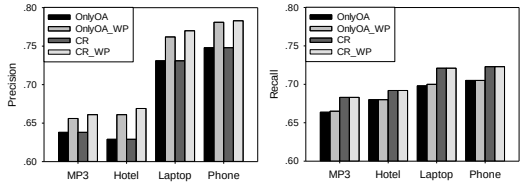
In this section, we try to prove the necessity of considering word preference in Eq.6. Besides the comparison between CR and CR_WP performed

in the main experiment in Section 4.2, we further incorporate word preference in aforementioned **OnlyOA**, named as **OnlyOA_WP**, which only employs opinion relations among words and equals to Eq.6 with $\lambda = 0$. Experimental results are shown in Figure 3. Because of space limitation, we only show the results of the same domains in section 4.3,

Form results, we observe that CR_WP outperforms CR, and OnlyOA_WP outperforms OnlyOA in all domains, especially on precision. These observations demonstrate that considering word preference is very important for opinion targets/words extraction. We believe the reason is that exploiting word preference can provide more fine information for opinion target/word candidates' confidence estimation. Thus the performance can be improved.



(a) Opinion Target Extraction Results



(b) Opinion Word Extraction Results

Figure 3: Experimental results when considering word preference

4.5 Parameter Sensitivity

In this subsection, we discuss the variation of extraction performance when changing λ and μ in Eq.6. Due to space limitation, we only show the F-measure of CR_WP on four domains. Experimental results are shown in Figure 4 and Figure 5. The left graphs in Figure 4 and 5 present the performance variation of CR_WP with varying λ from 0 to 0.9 and fixing $\mu = 0.1$. The right graphs in Figure 4 and 5 present the performance variation of CR_WP with varying μ from 0 to 0.6 and fixing $\lambda = 0.4$.

In the left graphs in Figure 4 and 5, we observe the best performance is obtained when $\lambda = 0.4$. It indicates that opinion relations and semantic relations are both useful for extracting opinion targets/words. The extraction performance is beneficial from their combination. In the right graphs in Figure 4 and 5, the best performance is obtained when $\mu = 0.1$. It indicates prior knowledge is useful for extraction. When μ increases, performance, however, decreases. It demonstrates that incorporating more prior knowledge into our algorithm would restrain other useful clues on estimating candidate confidence, and hurt the performance.

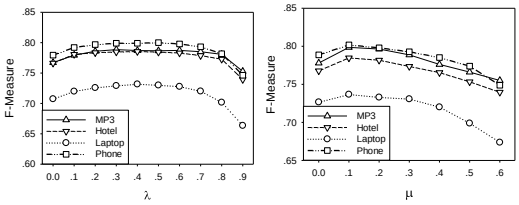


Figure 4: Opinion targets extraction results

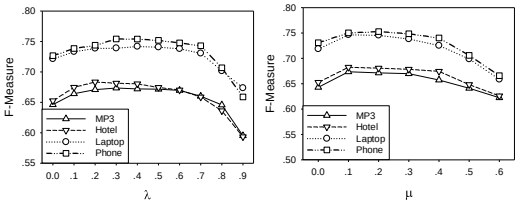


Figure 5: Opinion words extraction results

5 Conclusions

This paper presents a novel method with graph co-ranking to co-extract opinion targets/words. We model extracting opinion targets/words as a co-ranking process, where multiple heterogenous relations are modeled in a unified model to make co-operative effects on the extraction. In addition, we especially consider word preference in co-ranking process to perform more precise extraction. Compared to the state-of-the-art methods, experimental results prove the effectiveness of our method.

Acknowledgement

This work was sponsored by the National Basic Research Program of China (No. 2014CB340500), the National Natural Science Foundation of China (No. 61272332 and No. 61202329), the National High Technology Development 863 Program of China (No. 2012AA011102), and CCF-Tencent Open Research Fund.

References

Juergen Bross and Heiko Ehrig. 2013. Automatic construction of domain and aspect specific sentiment

lexicons for customer review mining. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, CIKM '13, pages 1077–1086, New York, NY, USA. ACM.

Zhen Hai, Kuiyu Chang, and Gao Cong. 2012. One seed to find them all: mining opinion features via association. In *CIKM*, pages 255–264.

Zhen Hai, Kuiyu Chang, Jung-Jae Kim, and Christopher C. Yang. 2013. Identifying features in opinion mining via intrinsic and extrinsic domain relevance. *IEEE Transactions on Knowledge and Data Engineering*, 99(PrePrints):1.

Mingqin Hu and Bing Liu. 2004a. Mining opinion features in customer reviews. In *Proceedings of Conference on Artificial Intelligence (AAAI)*.

Minqing Hu and Bing Liu. 2004b. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 168–177, New York, NY, USA. ACM.

Jon M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, September.

Fangtao Li, Chao Han, Minlie Huang, Xiaoyan Zhu, Yingju Xia, Shu Zhang, and Hao Yu. 2010. Structure-aware review mining and summarization. In Chu-Ren Huang and Dan Jurafsky, editors, *COLING*, pages 653–661. Tsinghua University Press.

Bing Liu, Minqing Hu, and Junsheng Cheng. 2005. Opinion observer: analyzing and comparing opinions on the web. In Allan Ellis and Tatsuya Hagino, editors, *WWW*, pages 342–351. ACM.

Kang Liu, Liheng Xu, and Jun Zhao. 2012. Opinion target extraction using word-based translation model. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1346–1356, Jeju Island, Korea, July. Association for Computational Linguistics.

Kang Liu, Liheng Xu, Yang Liu, and Jun Zhao. 2013a. Opinion target extraction using partially supervised word alignment model.

Kang Liu, Liheng Xu, and Jun Zhao. 2013b. Syntactic patterns versus word alignment: Extracting opinion targets from online reviews.

Tengfei Ma and Xiaojun Wan. 2010. Opinion target extraction in chinese news comments. In Chu-Ren Huang and Dan Jurafsky, editors, *COLING (Posters)*, pages 782–790. Chinese Information Processing Society of China.

Samaneh Moghaddam and Martin Ester. 2011. Ilda: Interdependent lda model for learning latent aspects and their ratings from online product reviews. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 665–674, New York, NY, USA. ACM.

Samaneh Moghaddam and Martin Ester. 2012a. Aspect-based opinion mining from product reviews. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 1184–1184, New York, NY, USA. ACM.

Samaneh Moghaddam and Martin Ester. 2012b. On the design of lda models for aspect-based opinion mining. In *CIKM*, pages 803–812.

Arjun Mukherjee and Bing Liu. 2012. Aspect extraction through semi-supervised modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 339–348, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ana-Maria Popescu and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 339–346, Stroudsburg, PA, USA. Association for Computational Linguistics.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Che. 2009. Expanding domain sentiment lexicon through double propagation.

Guang Qiu, Bing Liu 0001, Jiajun Bu, and Chun Chen. 2011. Opinion word expansion and target extraction through double propagation. *Computational Linguistics*, 37(1):9–27.

Qi Su, Xinying Xu, Honglei Guo, Zhili Guo, Xian Wu, Xiaoxun Zhang, Bin Swen, and Zhong Su. 2008. Hidden sentiment association in chinese web opinion mining. In Jinpeng Huai, Robin Chen, Hsiao-Wuen Hon, Yunhao Liu, Wei-Ying Ma, Andrew Tomkins, and Xiaodong Zhang 0001, editors, *WWW*, pages 959–968. ACM.

Bo Wang and Houfeng Wang. 2008. Bootstrapping both product features and opinion words from chinese customer reviews with cross-inducing.

Hongning Wang, Yue Lu, and ChengXiang Zhai. 2011. Latent aspect rating analysis without aspect keyword supervision. In Chid Apt, Joydeep Ghosh, and Padhraic Smyth, editors, *KDD*, pages 618–626. ACM.

Yuanbin Wu, Qi Zhang, Xuanjing Huang, and Lide Wu. 2009. Phrase dependency parsing for opinion mining. In *EMNLP*, pages 1533–1541. ACL.

Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long*

Papers), pages 1640–1649, Sofia, Bulgaria, August. Association for Computational Linguistics.

Lei Zhang, Bing Liu, Suk Hwan Lim, and Eamonn O'Brien-Strain. 2010. Extracting and ranking product features in opinion documents. In Chu-Ren Huang and Dan Jurafsky, editors, *COLING (Posters)*, pages 1462–1470. Chinese Information Processing Society of China.

Wayne Xin Zhao, Jing Jiang, Hongfei Yan, and Xiaoming Li. 2010. Jointly modeling aspects and opinions with a maxent-lda hybrid. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 56–65, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jingbo Zhu, Huizhen Wang, Benjamin K. Tsou, and Muhua Zhu. 2009. Multi-aspect opinion polling from textual reviews. In David Wai-Lok Cheung, Il-Yeol Song, Wesley W. Chu, Xiaohua Hu, and Jimmy J. Lin, editors, *CIKM*, pages 1799–1802. ACM.

Context-aware Learning for Sentence-level Sentiment Analysis with Posterior Regularization

Bishan Yang
Department of Computer Science
Cornell University
bishan@cs.cornell.edu

Claire Cardie
Department of Computer Science
Cornell University
cardie@cs.cornell.edu

Abstract

This paper proposes a novel context-aware method for analyzing sentiment at the level of individual sentences. Most existing machine learning approaches suffer from limitations in the modeling of complex linguistic structures across sentences and often fail to capture non-local contextual cues that are important for sentiment interpretation. In contrast, our approach allows structured modeling of sentiment while taking into account both local and global contextual information. Specifically, we encode intuitive lexical and discourse knowledge as expressive constraints and integrate them into the learning of conditional random field models via posterior regularization. The context-aware constraints provide additional power to the CRF model and can guide semi-supervised learning when labeled data is limited. Experiments on standard product review datasets show that our method outperforms the state-of-the-art methods in both the supervised and semi-supervised settings.

1 Introduction

The ability to extract sentiment from text is crucial for many opinion-mining applications such as opinion summarization, opinion question answering and opinion retrieval. Accordingly, extracting sentiment at the fine-grained level (e.g. at the sentence- or phrase-level) has received increasing attention recently due to its challenging nature and its importance in supporting these opinion analysis tasks (Pang and Lee, 2008).

In this paper, we focus on the task of sentence-level sentiment classification in online reviews. Typical approaches to the task employ supervised machine learning algorithms with rich features and take into account the interactions between words to handle compositional effects such as polarity reversal (e.g. (Nakagawa et al., 2010; Socher et al., 2013)). Still, their methods can encounter difficulty when the sentence on its own does not contain strong enough sentiment signals (due to the lack of statistical evidence or the requirement for background knowledge). Consider the following review for example,

> 1. Hearing the music in real stereo is a true revelation. 2. You can feel that the music is no longer constrained by the mono recording. 3. In fact, it is more like the players are performing on a stage in front of you ...

Existing feature-based classifiers may be effective in identifying the positive sentiment of the first sentence due to the use of the word *revelation*, but they could be less effective in the last two sentences due to the lack of explicit sentiment signals. However, if we examine these sentences within the discourse context, we can see that: the second sentence expresses sentiment towards the same aspect – *the music* – as the first sentence; the third sentence expands the second sentence with the discourse connective *In fact*. These discourse-level relations help indicate that sentence 2 and 3 are likely to have positive sentiment as well.

The importance of discourse for sentiment analysis has become increasingly recognized. Most existing work considers discourse relations between adjacent sentences or clauses and incorporates them as constraints (Kanayama and Nasukawa, 2006; Zhou et al., 2011) or features in classifiers Trivedi and Eisenstein (2013; Lazaridou et al. (2013). Very little work has explored long-distance discourse relations for sentiment analysis. Somasundaran et al. (2008) defines coreference relations on opinion targets and applies them to constrain the polarity of sentences.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 325–335,
Baltimore, Maryland, USA, June 23-25 2014.

However, the discourse relations were obtained from fine-grained annotations and implemented as hard constraints on polarity.

Obtaining sentiment labels at the fine-grained level is costly. Semi-supervised techniques have been proposed for sentence-level sentiment classification (Täckström and McDonald, 2011a; Qu et al., 2012). However, they rely on a large amount of document-level sentiment labels that may not be naturally available in many domains.

In this paper, we propose a sentence-level sentiment classification method that can (1) incorporate rich discourse information at both local and global levels; (2) encode discourse knowledge as soft constraints during learning; (3) make use of unlabeled data to enhance learning. Specifically, we use the Conditional Random Field (CRF) model as the learner for sentence-level sentiment classification, and incorporate rich discourse and lexical knowledge as soft constraints into the learning of CRF parameters via Posterior Regularization (PR) (Ganchev et al., 2010). As a framework for structured learning with constraints, PR has been successfully applied to many structural NLP tasks (Ganchev et al., 2009; Ganchev et al., 2010; Ganchev and Das, 2013). Our work is the first to explore PR for sentiment analysis. Unlike most previous work, we explore a rich set of structural constraints that cannot be naturally encoded in the feature-label form, and show that such constraints can improve the performance of the CRF model.

We evaluate our approach on the sentence-level sentiment classification task using two standard product review datasets. Experimental results show that our model outperforms state-of-the-art methods in both the supervised and semi-supervised settings. We also show that discourse knowledge is highly useful for improving sentence-level sentiment classification.

2 Related Work

There has been a large amount of work on sentiment analysis at various levels of granularity (Pang and Lee, 2008). In this paper, we focus on the study of sentence-level sentiment classification. Existing machine learning approaches for the task can be classified based on the use of two ideas. The first idea is to exploit sentiment signals at the sentence level by learning the relevance of sentiment and words while taking into account the context in which they occur: Nakagawa et al. (2010) uses tree-CRF to model word interactions based on dependency tree structures; Choi and Cardie (2008) applies compositional inference rules to handle polarity reversal; Socher et al. (2011) and Socher et al. (2013) compute compositional vector representations for words and phrases and use them as features in a classifier.

The second idea is to exploit sentiment signals at the inter-sentential level. Polanyi and Zaenen (2006) argue that discourse structure is important in polarity classification. Various attempts have been made to incorporate discourse relations into sentiment analysis: Pang and Lee (2004) explored the consistency of subjectivity between neighboring sentences; Mao and Lebanon (2007),McDonald et al. (2007), and Täckström and McDonald (2011a) developed structured learning models to capture sentiment dependencies between adjacent sentences; Kanayama and Nasukawa (2006) and Zhou et al. (2011) use discourse relations to constrain two text segments to have either the same polarity or opposite polarities; Trivedi and Eisenstein (2013) and Lazaridou et al. (2013) encode the discourse connectors as model features in supervised classifiers. Very little work has explored long-distance discourse relations. Somasundaran et al. (2008) define opinion target relations and apply them to constrain the polarity of text segments annotated with target relations. Recently, Zhang et al. (2013) explored the use of explanatory discourse relations as soft constraints in a Markov Logic Network framework for extracting subjective text segments.

Leveraging both ideas, our approach exploits sentiment signals from both intra-sentential and inter-sentential context. It has the advantages of utilizing rich discourse knowledge at different levels of context and encoding it as soft constraints during learning.

Our approach is also semi-supervised. Compared to the existing work on semi-supervised learning for sentence-level sentiment classification (Täckström and McDonald, 2011a; Täckström and McDonald, 2011b; Qu et al., 2012), our work does not rely on a large amount of coarse-grained (document-level) labeled data, instead, distant supervision mainly comes from linguistically-motivated constraints.

Our work also relates to the study of posterior regularization (PR) (Ganchev et al., 2010). PR has been successfully applied to many structured NLP

tasks such as dependency parsing, information extraction and cross-lingual learning tasks (Ganchev et al., 2009; Bellare et al., 2009; Ganchev et al., 2010; Ganchev and Das, 2013). Most previous work using PR mainly experiments with feature-label constraints. In contrast, we explore a rich set of linguistically-motivated constraints which cannot be naturally formulated in the feature-label form. We also show that constraints derived from the discourse context can be highly useful for disambiguating sentence-level sentiment.

3 Approach

In this section, we present the details of our proposed approach. We formulate the sentence-level sentiment classification task as a sequence labeling problem. The inputs to the model are sentence-segmented documents annotated with sentence-level sentiment labels (positive, negative or neutral) along with a set of unlabeled documents. During prediction, the model outputs sentiment labels for a sequence of sentences in the test document. We utilize conditional random fields and use Posterior Regularization (PR) to learn their parameters with a rich set of context-aware constraints.

In what follows, we first briefly describe the framework of Posterior Regularization. Then we introduce the context-aware constraints derived based on intuitive discourse and lexical knowledge. Finally we describe how to perform learning and inference with these constraints.

3.1 Posterior Regularization

PR is a framework for structured learning with constraints (Ganchev et al., 2010). In this work, we apply PR in the context of CRFs for sentence-level sentiment classification.

Denote $\mathbf{x}$ as a sequence of sentences within a document and $\mathbf{y}$ as a vector of sentiment labels associated with $\mathbf{x}$. The CRF model the following conditional probabilities:

$$p_\theta(\mathbf{y}|\mathbf{x}) = \frac{\exp(\theta \cdot f(\mathbf{x}, \mathbf{y}))}{Z_\theta(\mathbf{x})}$$

where $f(\mathbf{x}, \mathbf{y})$ are the model features, θ are the model parameters, and $Z_\theta(\mathbf{x}) = \sum_{\mathbf{y}} \exp(\theta \cdot f(\mathbf{x}, \mathbf{y}))$ is a normalization constant. The objective function for a standard CRF is to maximize the log-likelihood over a collection of labeled documents plus a regularization term:

$$\max_\theta \mathcal{L}(\theta) = \max_\theta \sum_{(\mathbf{x},\mathbf{y})} \log p_\theta(\mathbf{y}|\mathbf{x}) - \frac{||\theta||_2^2}{2\delta^2}$$

PR makes the assumption that the labeled data we have is not enough for learning good model parameters, but we have a set of constraints on the posterior distribution of the labels. We can define the set of desirable posterior distrbutions as

$$\mathcal{Q} = \{q(\mathbf{Y}) : E_q[\phi(\mathbf{X}, \mathbf{Y})] = \mathbf{b}\} \quad (1)$$

where ϕ is a constraint function, $\mathbf{b}$ is a vector of desired values of the expectations of the constraint functions under the distribution q [1]. Note that the distribution q is defined over a collection of unlabeled documents where the constraint functions apply, and we assume independence between documents.

The PR objective can be written as the original model objective penalized with a regularization term, which minimizes the KL-divergence between the desired model posteriors and the learned model posteriors with an L2 penalty [2] for the constraint violations.

$$\begin{aligned}\max_\theta \mathcal{L}(\theta) - \min_{q\in\mathcal{Q}} \{KL(q(\mathbf{Y})||p_\theta(\mathbf{Y}|\mathbf{X})) \\ + \beta||E_q[\phi(\mathbf{X}, \mathbf{Y})] - \mathbf{b}||_2^2\}\end{aligned} \quad (2)$$

The objective can be optimized by an EM-like scheme that iteratively solves the minimization problem and the maximization problem. Solving the minimization problem is equivalent to solving its dual since the objective is convex. The dual problem is

$$\arg\max_\lambda \lambda \cdot \mathbf{b} - \log Z_\lambda(X) - \frac{1}{4\beta}||\lambda||_2^2 \quad (3)$$

We optimize the objective function 2 using stochastic projected gradient, and compute the learning rate using AdaGrad (Duchi et al., 2010).

3.2 Context-aware Posterior Constraints

We develop a rich set of context-aware posterior constraints for sentence-level sentiment analysis by exploiting lexical and discourse knowledge. Specifically, we construct the lexical constraints by extracting sentiment-bearing patterns

[1] In general, inequality constraints can also be used. We focus on the equality constraints since we found them to express the sentiment-relevant constraints well.

[2] Other convex functions can be used for the penalty. We use L2 norm because it works well in practice. β is a regularization constant

within sentences and construct the discourse-level constraints by extracting discourse relations that indicate sentiment coherence or sentiment changes both within and across sentences. Each constraint can be formulated as equality between the expectation of a constraint function value and a desired value set by prior knowledge. The equality is not strictly enforced (due to the regularization in the PR objective 2). Therefore all the constraints are applied as soft constraints. Table 1 provides intuitive description and examples for all the constraints used in our model.

Lexical Patterns The existence of a polarity-carrying word alone may not correctly indicate the polarity of the sentence, as the polarity can be reversed by other polarity-reversing words. We extract lexical patterns that consist of polar words and negators [3], and apply the heuristics based on compositional semantics (Choi and Cardie, 2008) to assign a sentiment value to each pattern.

We encode the extracted lexical patterns along with their sentiment values as feature-label constraints. The constraint function can be written as

$$\phi_w(x, y) = \sum_i f_w(x_i, y_i)$$

where $f_w(x_i, y_i)$ is a feature function which has value 1 when sentence x_i contains the lexical pattern w and its sentiment label y_i equals to the expected sentiment value and has value 0 otherwise. The constraint expectation value is set to be the prior probability of associating w with its sentiment value. Note that sentences with neutral sentiment can also contain such lexical patterns. Therefore we allow the lexical patterns to be assigned a neutral sentiment with a prior probability r_0 (we compute this value as the empirical probability of neutral sentiment in the training documents). Using the polarity indicated by lexical patterns to constrain the sentiment of sentences is quite aggressive. Therefore we only consider lexical patterns that are strongly discriminative (many opinion words in the lexicon only indicate sentiment with weak strength). The selected lexical patterns include a handful of seed patterns (such as "pros" and "cons") and the lexical patterns that have high precision (larger then 0.9) of predicting sentiment in the training data.

[3]The polar words are identified using the MPQA lexicon and the negators are identified using a handful of seed words extended by the General Inquirer dictionary and WordNet as described in (Choi and Cardie, 2008).

Discourse Connectives. Lexical patterns can be limited in capturing contextual information since they only look at interactions between words within an expression. To capture context at the clause or sentence level, we consider discourse connectives, which are cue phrases or words that indicate discourse relations between adjacent sentences or clauses. To identify discourse connectives, we apply a discourse tagger trained on the Penn Discourse Treebank (Prasad et al., 2008) [4] to our data. Discourse connectives are tagged with four senses: *Expansion, Contingency, Comparison, Temporal*.

Discourse connectives can operate at both intra-sentential and inter-sentential level. For example, the word "although" is often used to connect two polar clauses within a sentence, while the word "however" is often used to at the beginning of the sentence to connect two polar sentences. It is important to distinguish these two types of discourse connectives. We consider a discourse connective to be intra-sentential if it has the *Comparison* sense and connects two polar clauses with opposite polarities (determined by the lexical patterns). We construct a feature-label constraint for each intra-sentential discourse connective and set its expected sentiment value to be neutral.

Unlike the intra-sentential discourse connectives, the inter-sentential discourse connectives can indicate sentiment transitions between sentences. Intuitively, discourse connectives with the senses of *Expansion* (e.g. also, for example, furthermore) and *Contingency* (e.g. as a result, hence, because) are likely to indicate sentiment coherence; discourse connectives with the sense of *Comparison* (e.g. but, however, nevertheless) are likely to indicate sentiment changes. This intuition is reasonable but it assumes the two sentences connected by the discourse connective are both polar sentences. In general, discourse connectives can also be used to connect non-polar (neutral) sentences. Thus it is hard to directly constrain the posterior expectation for each type of sentiment transitions using inter-sentential discourse connectives.

Instead, we impose constraints on the model posteriors by reducing constraint violations. We

[4]http://www.cis.upenn.edu/~epitler/discourse.html

Types	Description and Examples	Inter-sentential
Lexical patterns	The sentence containing a polar lexical pattern w tends to have the polarity indicated by w. Example lexical patterns are *annoying, hate, amazing, not disappointed, no concerns, favorite, recommend*.	
Discourse Connectives (clause)	The sentence containing a discourse connective c which connects its two clauses that have opposite polarities indicated by the lexical patterns tends to have neutral sentiment. Example connectives are *while, although, though, but*.	
Discourse Connectives (sentence)	Two adjacent sentences which are connected by a discourse connective c tends to have the same polarity if c indicates a *Expansion* or *Contingency* relation, e.g. *also, for example, in fact, because* ; opposite polarities if c indicates a *Comparison* relation, e.g. *otherwise, nevertheless, however*.	✓
Coreference	The sentences which contain coreferential entities appeared as targets of opinion expressions tend to have the same polarity.	✓
Listing patterns	A series of sentences connected via a listing tend to have the same polarity.	✓
Global labels	The sentence-level polarity tends to be consistent with the document-level polarity.	✓

Table 1: Summarization of Posterior Constraints for Sentence-level Sentiment Classification

define the following constraint function:

$$\phi_{c,s}(x, y) = \sum_i f_{c,s}(x_i, y_i, y_{i-1})$$

where c denotes a discourse connective, s indicates its sense, and $f_{c,s}$ is a penalty function that takes value 1.0 when y_i and y_{i-1} form a contradictory sentiment transition, that is, $y_i \neq_{polar} y_{i-1}$ if $s \in \{\textit{Expansion}, \textit{Contingency}\}$, or $y_i =_{polar} y_{i-1}$ if $s = \textit{Comparison}$. The desired value for the constraint expectation is set to 0 so that the model is encouraged to have less constraint violations.

Opinion Coreference Sentences in a discourse can be linked by many types of coherence relations (Jurafsky et al., 2000). Coreference is one of the commonly used relations in written text. In this work, we explore coreference in the context of sentence-level sentiment analysis. We consider a set of polar sentences to be linked by the *opinion coreference* relation if they contain coreferring opinion-related entities. For example, the following sentences express opinions towards "the speaker phone", "The speaker phone" and "it" respectively. As these opinion targets are coreferential (referring to the same entity "the speaker phone"), they are linked by the *opinion coreference* relation [5].

> My favorite features are **the speaker phone** and the radio. **The speaker phone** is very functional. I use **it** in the car, very audible even with freeway noise.

[5]In general, the opinion-related entities include both the opinion targets and the opinion holders. In this work, we only consider the targets since we experiment with single-author product reviews. The opinion holders can be included in a similar way as the opinion targets.

Our coreference relations indicated by opinion targets overlap with the *same target* relation introduced in (Somasundaran et al., 2009). The differences are: (1) we encode the coreference relations as soft constraints during learning instead of applying them as hard constraints during inference time; (2) our constraints can apply to both polar and non-polar sentences; (3) our identification of coreference relations is automatic without any fine-grained annotations for opinion targets.

To extract coreferential opinion targets, we apply Stanford's coreference system (Lee et al., 2013) to extract coreferential mentions in the document, and then apply a set of syntactic rules to identify opinion targets from the extracted mentions. The syntactic rules correspond to the shortest dependency paths between an opinion word and an extracted mention. We consider the 10 most frequent dependency paths in the training data. Example dependency paths include *nsubj*(opinion, mention), *nobj*(opinion, mention), and *amod*(mention, opinion).

For sentences connected by the opinion coreference relation, we expect their sentiment to be consistent. To encode this intuition, we define the following constraint function:

$$\phi_{coref}(x, y) = \sum_{i, ant(i)=j, j \geq 0} f_{coref}(x_i, x_j, y_i, y_j)$$

where $ant(i)$ denotes the index of the sentence which contains an antecedent target of the target mentioned in sentence i (the antecedent relations over pairs of opinion targets can be constructed using the coreference resolver), and f_{coref} is a penalty function which takes value 1.0 when the expected sentiment coherency is violated, that is, $y_i \neq_{polar} y_j$. Similar to the inter-sentential dis-

course connectives, modeling opinion coreference via constraint violations allows the model to handle neutral sentiment. The expected value of the constraint functions is set to 0.

Listing Patterns Another type of coherence relations we observe in online reviews is listing, where a reviewer expresses his/her opinions by listing a series of statements followed by a sequence of numbers. For example, "1. It's smaller than the ipod mini .... 2. It has a removable battery ....". We expect sentences connected by a listing to have consistent sentiment. We implement this constraint in the same form as the coreference constraint (the antecedent assignments are constructed from the numberings).

Global Sentiment Previous studies have demonstrated the value of document-level sentiment in guiding the semi-supervised learning of sentence-level sentiment (Täckström and McDonald, 2011b; Qu et al., 2012). In this work, we also take into account this information and encode it as posterior constraints. Note that these constraints are not necessary for our model and can be applied when the document-level sentiment labels are naturally available.

Based on an analysis of the Amazon review data, we observe that sentence-level sentiment usually doesn't conflict with the document-level sentiment in terms of polarity. For example, the proportion of negative sentences in the positive documents is very small compared to the proportion of positive sentences. To encode this intuition, we define the following constraint function:

$$\phi_g(x, y) = \sum_i^n \delta(y_i \neq_{polar} g)/n$$

where $g \in \{positive, negative\}$ denotes the sentiment value of a polar document, n is the total number of sentences in x, and δ is an indicator function. We hope the expectation of the constraint function takes a small value. In our experiments, we set the expected value to be the empirical estimate of the probability of "conflicting" sentiment in polar documents using the training data.

3.3 Training and Inference

During training, we need to compute the constraint expectations and the feature expectations under the auxiliary distribution q at each gradient step. We can derive q by solving the dual problem in 3:

$$q(\mathbf{y}|\mathbf{x}) = \frac{exp(\theta \cdot f(\mathbf{x}, \mathbf{y}) + \lambda \cdot \phi(\mathbf{x}, \mathbf{y}))}{Z_{\lambda,\theta}(X)} \quad (4)$$

where $Z_{\lambda,\theta}(X)$ is a normalization constant. Most of our constraints can be factorized in the same way as factorizing the model features in the first-order CRF model, and we can compute the expectations under q very efficiently using the forward-backward algorithm. However, some of our discourse constraints (opinion coreference and listing) can break the tractable structure of the model. For constraints with higher-order structures, we use Gibbs Sampling (Geman and Geman, 1984) to approximate the expectations. Given a sequence $\mathbf{x}$, we sample a label $\mathbf{y}_i$ at each position i by computing the unnormalized conditional probabilities $p(\mathbf{y}_i = l|\mathbf{y}_{-i}) \propto exp(\theta \cdot f(\mathbf{x}, \mathbf{y}_i = l, \mathbf{y}_{-i}) + \lambda \cdot \phi(\mathbf{x}, \mathbf{y}_i = l, \mathbf{y}_{-i}))$ and renormalizing them. Since the possible label assignments only differ at position i, we can make the computation efficient by maintaining the structure of the coreference clusters and precomputing the constraint function for different types of violations.

During inference, we find the best label assignment by computing $\arg\max_{\mathbf{y}} q(\mathbf{y}|\mathbf{x})$. For documents where the higher-order constraints apply, we use the same Gibbs sampler as described above to infer the most likely label assignment, otherwise, we use the Viterbi algorithm.

4 Experiments

We experimented with two product review datasets for sentence-level sentiment classification: the Customer Review (CR) data (Hu and Liu, 2004)[6] which contains 638 reviews of 14 products such as cameras and cell phones, and the Multi-domain Amazon (MD) data from the test set of Täckström and McDonald (2011a) which contains 294 reivews from 5 different domains. As in Qu et al. (2012), we chose the books, electronics and music domains for evaluation. Each domain also comes with 33,000 extra reviews with only document-level sentiment labels.

We evaluated our method in two settings: supervised and semi-supervised. In the supervised setting, we treated the test data as unlabeled data and performed transductive learning. In the semi-supervised setting, our unlabeled data consists of

[6]Available at `http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html`.

both the available unlabeled data and the test data. For each domain in the MD dataset, we made use of no more than 100 unlabeled documents in which our posterior constraints apply. We adopted the evaluation schemes used in previous work: 10-fold cross validation for the CR dataset and 3-fold cross validation for the MD dataset. We also report both two-way classification (positive vs. negative) and three-way classification results (positive, negative or neutral). We use accuracy as the performance measure. In our tables, boldface numbers are statistically significant by paired t-test for $p < 0.05$ against the best baseline developed in this paper [7].

We trained our model using a CRF incorporated with the proposed posterior constraints. For the CRF features, we include the tokens, the part-of-speech tags, the prior polarities of lexical patterns indicated by the opinion lexicon and the negator lexicon, the number of positive and negative tokens and the output of the vote-flip algorithm (Choi and Cardie, 2009). In addition, we include the discourse connectives as local or transition features and the document-level sentiment labels as features (only available in the MD dataset).

We set the CRF regularization parameter $\sigma = 1$ and set the posterior regularization parameter β and γ (a trade-off parameter we introduce to balance the supervised objective and the posterior regularizer in 2) by using grid search [8]. For approximation inference with higher-order constraints, we perform 2000 Gibbs sampling iterations where the first 1000 iterations are burn-in iterations. To make the results more stable, we construct three Markov chains that run in parallel, and select the sample with the largest objective value.

All posterior constraints were developed using the training data on each training fold. For the MD dataset, we also used the dvd domain as additional labeled data for developing the constraints.

Baselines. We compared our method to a number of baselines: (1) CRF: CRF with the same set of model features as in our method. (2) CRF-INF: CRF augmented with inference constraints. We can incorporate the proposed constraints (constraints derived from lexical patterns and discourse connectives) as hard constraints into CRF during inference by manually setting λ in equation 4 to a large value,[9]. When λ is large enough, it is equivalent to adding hard constraints to the viterbi inference. To better understand the different effects of lexical and discourse constraints, we report results for applying only the lexical constraints (CRF-INF$_{lex}$) as well as results for applying only the discourse constraints (CRF-INF$_{disc}$). (3) PR$_{lex}$: a variant of our PR model which only applies the lexical constraints. For the three-way classification task on the MD dataset, we also implemented the following baselines: (4) VOTEFLIP: a rule-based algorithm that leverages the positive, negative and neutral cues along with the effect of negation to determine the sentence sentiment (Choi and Cardie, 2009). (5) DOCORACLE: assigns each sentence the label of its corresponding document.

[7] Significance test was not conducted over the previous methods as we do not have their results for each fold.

[8] We conducted 10-fold cross-validation on each training fold with the parameter space: $\beta : [0.01, 0.05, 0.1, 0.5, 1.0]$ and $\gamma : [0.1, 0.5, 1.0, 5.0, 10.0]$.

Methods	CR	MD
CRF	81.1	67.0
CRF-inf$_{lex}$	80.9	66.4
CRF-inf$_{disc}$	81.1	67.2
PR$_{lex}$	81.8	69.7
PR	**82.7**	**70.6**
Previous work		
TreeCRF (Nakagawa et al., 2010)	81.4	-
Dropout LR (Wang and Manning, 2013)	82.1	-

Table 2: Accuracy results (%) for supervised sentiment classification (two-way)

	Books	Electronics	Music	Avg
VoteFlip	44.6	45.0	47.8	45.8
DocOracle	53.6	50.5	63.0	55.7
CRF	57.4	57.5	61.8	58.9
CRF-inf$_{lex}$	56.7	56.4	60.4	57.8
CRF-inf$_{disc}$	57.2	57.6	62.1	59.0
PR$_{lex}$	60.3	59.9	63.2	61.1
PR	**61.6**	**61.0**	**64.4**	**62.3**
Previous work				
HCRF	55.9	61.0	58.7	58.5
MEM	59.7	59.6	63.8	61.0

Table 3: Accuracy results (%) for semi-supervised sentiment classification (three-way) on the MD dataset

4.1 Results

We first report results on a binary (positive or negative) sentence-level sentiment classification task. For this task, we used the supervised setting and performed transductive learning for our model. Table 2 shows the accuracy results. We can see

[9] We set λ to 1000 for the lexical constraints and -1000 to the discourse connective constraints in the experiments

	Books pos/neg/neu	**Electronics** pos/neg/neu	**Music** pos/neg/neu
VoteFlip	43/42/47	45/46/44	50/46/46
DocOracle	54/60/49	57/54/42	72/65/52
CRF	47/51/64	60/61/52	67/60/58
CRF-inf$_{lex}$	46/52/63	59/61/50	65/59/57
CRF-inf$_{disc}$	47/51/64	60/61/52	67/61/59
PR$_{lex}$	50/56/66	64/63/53	67/64/59
PR	52/56/68	64/66/53	69/65/60

Table 4: F1 scores for each sentiment category (positive, negative and neutral) for semi-supervised sentiment classification on the MD dataset

that PR significantly outperforms all other baselines in both the CR dataset and the MD dataset (average accuracy across domains is reported). The poor performance of CRF-INF$_{lex}$ indicates that directly applying lexical constraints as hard constraints during inference could only hurt the performance. CRF-INF$_{disc}$ slightly outperforms CRF but the improvement is not significant. In contrast, both PR$_{lex}$ and PR significantly outperform CRF, which implies that incorporating lexical and discourse constraints as posterior constraints is much more effective. The superior performance of PR over PR$_{lex}$ further suggests that the proper use of discourse information can significantly improve accuracy for sentence-level sentiment classification.

We also analyzed the model's performance on a three-way sentiment classification task. By introducing the "neutral" category, the sentiment classification problem becomes harder. Table 4 shows the results in terms of accuracy for each domain in the MD dataset. We can see that both PR and PR$_{lex}$ significantly outperform all other baselines in all domains. The rule-based baseline VOTEFLIP gave the weakest performance because it has no prediction power on sentences with no opinion words. DOCORACLE performs much better than VOTEFLIP and performs especially well on the *Music* domain. This indicates that the document-level sentiment is a very strong indicator of the sentence-level sentiment label. For the CRF baseline and its invariants, we observe a similar performance trend as in the two-way classification task: there is nearly no performance improvement from applying the lexical and discourse-connective-based constraints during CRF inference. In contrast, both PR$_{lex}$ and PR provide substantial improvements over CRF. This confirms that encoding lexical and discourse knowledge as posterior constraints allows the feature-based model to gain additional learning power for sentence-level sentiment prediction. In particular, incorporating discourse constraints leads to consistent improvements to our model. This demonstrates that our modeling of discourse information is effective and that taking into account the discourse context is important for improving sentence-level sentiment analysis. We also compare our results to the previously published results on the same dataset. HCRF (Täckström and McDonald, 2011a) and MEM (Qu et al., 2012) are two state-of-the-art semi-supervised methods for sentence-level sentiment classification. We can see that our best model PR gives the best results in most categories.

Table 4 shows the results in terms of F1 scores for each sentiment category (positive, negative and neutral). We can see that the PR models are able to provide improvements over all the sentiment categories compared to all the baselines in general. We observe that the DOCORACLE baseline provides very strong F1 scores on the positive and negative categories especially in the Books and Music domains, but very poor F1 on the neutral category. This is because it over-predicts the polar sentences in the polar documents, and predicts no polar sentences in the neutral documents. In contrast, our PR models provide more balanced F1 scores among all the sentiment categories. Compared to the CRF baseline and its variants, we found that the PR models can greatly improve the precision of predicting positive and negative sentences, resulting in a significant improvement on the positive/negative F1 scores. However, the improvement on the neutral category is modest. A plausible explanation is that most of our constraints focus on discriminating polar sentences. They can help reduce the errors of misclassifying polar sentences, but the model needs more constraints in order to distinguish neutral sentences from polar sentences. We plan to address this issue in future work.

4.2 Discussion

We analyze the errors to better understand the merits and limitations of the PR model. We found that the PR model is able to correct many CRF errors caused by the lack of labeled data. The first row in Table 5 shows an example of such errors.

Example Sentences	CRF	PR
Example 1: ⟨neg⟩ If I could, I would like to return it or exchange for something better.⟨/neg⟩	⟨neu⟩ ×	✓
Example 2: ⟨neg⟩ Things I wasn't a fan of – the ending was to cutesy for my taste.⟨/neg⟩ ⟨neg⟩ Also, all of the side characters (particularly the mom, vee, and the teacher) were incredibly flat and stereotypical to me.⟨/neg⟩	⟨neu⟩ ⟨pos⟩ ×	✓
Example 3: ⟨neg⟩ I also have excessive noise when I talk and have phone in my pocket while walking.⟨/neg⟩ ⟨neu⟩ But other models are no better.⟨/neu⟩	⟨neg⟩ ⟨pos⟩ ×	⟨neg⟩ ⟨pos⟩ ×

Table 5: Example sentences where PR succeeds and fails to correct the mistakes of CRF

The lexical features *return* and *exchange* may be good indicators of negative sentiment for the sentence. However, with limited labeled data, the CRF learner can only associate very weak sentiment signals to these features. In contrast, the PR model is able to associate stronger sentiment signals to these features by leveraging unlabeled data for indirect supervision. A simple lexicon-based constraint during inference time may also correct this case. However, hard-constraint baselines can hardly improve the performance in general because the contributions of different constraints are not learned and their combination may not lead to better predictions. This is also demonstrated by the limited performance of CRF-INF in our experiments.

We also found that the discourse constraints play an important role in improving the sentiment prediction. The lexical constraints alone are often not sufficient since their coverage is limited by the sentiment lexicon and they can only constrain sentiment locally. On the contrary, discourse constraints are not dependent on sentiment lexicons, and more importantly, they can provide sentiment preferences on multiple sentences at the same time. When combining discourse constraints with features from different sentences, the PR model becomes more powerful in disambiguating sentiment. The second example in Table 5 shows that the PR model learned with discourse constraints correctly predicts the sentiment of two sentences where no lexical constraints apply.

However, discourse constraints are not always helpful. One reason is that they do not constrain the neutral sentiment. As a result they could not help disambiguate neutral sentiment from polar sentiment, such as the third example in Table 5. This is also a problem for most of our lexical constraints. In general, it is hard to learn reliable indicators for the neutral sentiment. In the MD dataset, a neutral label may be given because the sentence contains mixed sentiment or no sentiment or it is off-topic. We plan to explore more refined constraints that can deal with the neutral sentiment in future work. Another limitation of the discourse constraints is that they could be affected by the errors of the discourse parser and the coreference resolver. A potential way to address this issue is to learn discourse constraints jointly with sentiment. We plan to study this in future research.

5 Conclusion

In this paper, we propose a context-aware approach for learning sentence-level sentiment. Our approach incorporates intuitive lexical and discourse knowledge as expressive constraints while training a conditional random field model via posterior regularization. We explore a rich set of context-aware constraints at both intra- and inter-sentential levels, and demonstrate their effectiveness in the analysis of sentence-level sentiment. While we focus on the sentence-level task, our approach can be easily extended to handle sentiment analysis at finer levels of granularity. Our experiments show that our model achieves better accuracy than existing supervised and semi-supervised models for the sentence-level sentiment classification task.

Acknowledgments

This work was supported in part by DARPA-BAA-12-47 DEFT grant #12475008 and NSF grant BCS-0904822. We thank Igor Labutov for helpful discussion and suggestions; Oscar Täckström and Lizhen Qu for providing their Amazon review datasets; and the anonymous reviewers for helpful comments and suggestions.

References

Kedar Bellare, Gregory Druck, and Andrew McCallum. 2009. Alternating projections for learning

with expectation constraints. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 43–50. AUAI Press.

Yejin Choi and Claire Cardie. 2008. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 793–801. Association for Computational Linguistics.

Yejin Choi and Claire Cardie. 2009. Adapting a polarity lexicon using integer linear programming for domain-specific sentiment classification. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 590–598. Association for Computational Linguistics.

John Duchi, Elad Hazan, and Yoram Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Kuzman Ganchev and Dipanjan Das. 2013. Cross-lingual discriminative learning of sequence models with posterior regularization.

Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proceedings of the ACL-IJCNLP*, pages 369–377.

Kuzman Ganchev, Joao Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *The Journal of Machine Learning Research*, 99:2001–2049.

Stuart Geman and Donald Geman. 1984. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):721–741.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Dan Jurafsky, James H Martin, Andrew Kehler, Keith Vander Linden, and Nigel Ward. 2000. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*, volume 2. MIT Press.

Hiroshi Kanayama and Tetsuya Nasukawa. 2006. Fully automatic lexicon expansion for domain-oriented sentiment analysis. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 355–363. Association for Computational Linguistics.

Angeliki Lazaridou, Ivan Titov, and Caroline Sporleder. 2013. A bayesian model for joint unsupervised induction of sentiment, aspect and discourse representations. In *To Appear in Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics*, Sofia, Bulgaria, August. Association for Computational Linguistics.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules.

Yi Mao and Guy Lebanon. 2007. Isotonic conditional random fields and local sentiment flow. *Advances in neural information processing systems*, 19:961.

Ryan McDonald, Kerry Hannan, Tyler Neylon, Mike Wells, and Jeff Reynar. 2007. Structured models for fine-to-coarse sentiment analysis. In *Annual Meeting-Association For Computational Linguistics*, volume 45, page 432.

Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. 2010. Dependency tree-based sentiment classification using crfs with hidden variables. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 786–794. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2008. *Opinion mining and sentiment analysis*. Now Pub.

Livia Polanyi and Annie Zaenen. 2006. Contextual valence shifters. In *Computing attitude and affect in text: Theory and applications*, pages 1–10. Springer.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. 2008. The penn discourse treebank 2.0. In *LREC*. Citeseer.

Lizhen Qu, Rainer Gemulla, and Gerhard Weikum. 2012. A weakly supervised model for sentence-level semantic orientation analysis with multiple experts. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 149–159. Association for Computational Linguistics.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics.

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

Swapna Somasundaran, Janyce Wiebe, and Josef Ruppenhofer. 2008. Discourse level opinion interpretation. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 801–808. Association for Computational Linguistics.

Swapna Somasundaran, Galileo Namata, Janyce Wiebe, and Lise Getoor. 2009. Supervised and unsupervised methods in employing discourse relations for improving opinion polarity classification. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 170–179. Association for Computational Linguistics.

Oscar Täckström and Ryan McDonald. 2011a. Discovering fine-grained sentiment with latent variable structured prediction models. In *Advances in Information Retrieval*, pages 368–374. Springer.

Oscar Täckström and Ryan McDonald. 2011b. Semi-supervised latent variable models for sentence-level sentiment analysis.

Rakshit Trivedi and Jacob Eisenstein. 2013. Discourse connectors for latent subjectivity in sentiment analysis. In *Proceedings of NAACL-HLT*, pages 808–813.

Sida Wang and Christopher Manning. 2013. Fast dropout training. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 118–126.

Qi Zhang, Jin Qian, Huan Chen, Jihua Kang, and Xuanjing Huang. 2013. Discourse level explanatory relation extraction from product reviews using first-order logic.

Lanjun Zhou, Binyang Li, Wei Gao, Zhongyu Wei, and Kam-Fai Wong. 2011. Unsupervised discovery of discourse relations for eliminating intra-sentence polarity ambiguities. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 162–171. Association for Computational Linguistics.

Product Feature Mining: Semantic Clues versus Syntactic Constituents

Liheng Xu, Kang Liu, Siwei Lai and Jun Zhao
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China
{lhxu, kliu, swlai, jzhao}@nlpr.ia.ac.cn

Abstract

Product feature mining is a key subtask in fine-grained opinion mining. Previous works often use syntax constituents in this task. However, syntax-based methods can only use discrete contextual information, which may suffer from data sparsity. This paper proposes a novel product feature mining method which leverages lexical and contextual semantic clues. Lexical semantic clue verifies whether a candidate term is related to the target product, and contextual semantic clue serves as a *soft* pattern miner to find candidates, which exploits semantics of each word in context so as to alleviate the data sparsity problem. We build a semantic similarity graph to encode lexical semantic clue, and employ a convolutional neural model to capture contextual semantic clue. Then Label Propagation is applied to combine both semantic clues. Experimental results show that our semantics-based method significantly outperforms conventional syntax-based approaches, which not only mines product features more accurately, but also extracts more infrequent product features.

1 Introduction

In recent years, opinion mining has helped customers a lot to make informed purchase decisions. However, with the rapid growth of e-commerce, customers are no longer satisfied with the overall opinion ratings provided by traditional sentiment analysis systems. The detailed functions or attributes of products, which are called *product features*, receive more attention. Nevertheless, a product may have thousands of features, which makes it impractical for a customer to investigate them all. Therefore, mining product features automatically from online reviews is shown to be a key step for opinion summarization (Hu and Liu, 2004; Qiu et al., 2009) and fine-grained sentiment analysis (Jiang et al., 2011; Li et al., 2012).

Previous works often mine product features via **syntactic constituent matching** (Popescu and Etzioni, 2005; Qiu et al., 2009; Zhang et al., 2010). The basic idea is that reviewers tend to comment on product features in similar syntactic structures. Therefore, it is natural to mine product features by using syntactic patterns. For example, in Figure 1, the upper box shows a dependency tree produced by Stanford Parser (de Marneffe et al., 2006), and the lower box shows a common syntactic pattern from (Zhang et al., 2010), where <feature/NN> is a wildcard to be fit in reviews and *NN* denotes the required POS tag of the wildcard. Usually, the product name *mp3* is specified, and when *screen* matches the wildcard, it is likely to be a product feature of *mp3*.

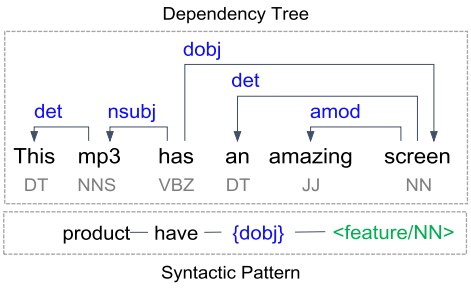


Figure 1: An example of syntax-based product feature mining procedure. The word *screen* matches the wildcard <feature/NN>. Therefore, *screen* is likely to be a product feature of *mp3*.

Generally, such syntactic patterns extract product features well but they still have some limitations. For example, the *product-have-feature* pattern may fail to find the *fm tuner* in a very similar case in Example 1(a), where the product is mentioned by using *player* instead of *mp3*. Similarly, it may also fail on Example 1(b), just with *have* replaced by *support*. In essence, syntactic pattern is

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 336–346,
Baltimore, Maryland, USA, June 23-25 2014.

a kind of *one-hot representation* for encoding the contexts, which can only use partial and discrete features, such as some key words (e.g., *have*) or shallow information (e.g., POS tags). Therefore, such a representation often suffers from the data sparsity problem (Turian et al., 2010).

One possible solution for this problem is using a more general pattern such as *NP-VB-feature*, where *NP* represents a noun or noun phrase and *VB* stands for any verb. However, this pattern becomes too general that it may find many irrelevant cases such as the one in Example 1(c), which is not talking about the product. Consequently, it is very difficult for a pattern designer to balance between precision and generalization.

Example 1:
(a) *This player has an fm tuner.*
(b) *This mp3 supports wma file.*
(c) *This review has helped people a lot.*
(d) *This mp3 has some flaws.*

To solve the problems stated above, it is argued that deeper semantics of contexts shall be exploited. For example, we can try to automatically discover that the verb *have* indicates a part-whole relation (Zhang et al., 2010) and *support* indicates a product-function relation, so that both *sth. have* and *sth. support* suggest that terms following them are product features, where *sth.* can be replaced by any terms that refer to the target product (e.g., *mp3*, *player*, etc.). This is called **contextual semantic clue**. Nevertheless, only using contexts is not sufficient enough. As in Example 1(d), we can see that the word *flaws* follows *mp3 have*, but it is not a product feature. Thus, a noise term may be extracted even with high contextual support. Therefore, we shall also verify whether a candidate is really related to the target product. We call it **lexical semantic clue**.

This paper proposes a novel bootstrapping approach for product feature mining, which leverages both semantic clues discussed above. Firstly, some reliable product feature seeds are automatically extracted. Then, based on the assumption that terms that are more semantically similar to the seeds are more likely to be product features, a graph which measures semantic similarities between terms is built to capture lexical semantic clue. At the same time, a semi-supervised convolutional neural model (Collobert et al., 2011) is employed to encode contextual semantic clue. Finally, the two kinds of semantic clues are combined by a Label Propagation algorithm.

In the proposed method, words are represented by continuous vectors, which capture latent semantic factors of the words (Turian et al., 2010). The vectors can be unsupervisedly trained on large scale corpora, and words with similar semantics will have similar vectors. This enables our method to be less sensitive to lexicon change, so that the data sparsity problem can be alleviated . The contributions of this paper include:

- It uses semantics of words to encode contextual clues, which exploits deeper level information than syntactic constituents. As a result, it mines product features more accurately than syntax-based methods.
- It exploits semantic similarity between words to capture lexical clues, which is shown to be more effective than co-occurrence relation between words and syntactic patterns. In addition, experiments show that the semantic similarity has the advantage of mining infrequent product features, which is crucial for this task. For example, one may say "*This hotel has low water pressure*", where *low water pressure* is seldom mentioned, but fatal to someone's taste.
- We compare the proposed semantics-based approach with three state-of-the-art syntax-based methods. Experiments show that our method achieves significantly better results.

The rest of this paper is organized as follows. Section 2 introduces related work. Section 3 describes the proposed method in details. Section 4 gives the experimental results. Lastly, we conclude this paper in Section 5.

2 Related Work

In product feature mining task, Hu and Liu (2004) proposed a pioneer research. However, the association rules they used may potentially introduce many noise terms. Based on the observation that product features are often commented on by similar syntactic structures, it is natural to use patterns to capture common syntactic constituents around product features.

Popescu and Etzioni (2005) designed some syntactic patterns to search for product feature candidates and then used Pointwise Mutual Information (PMI) to remove noise terms. Qiu et al. (2009) proposed eight heuristic syntactic rules to jointly extract product features and sentiment lexicons, where a bootstrapping algorithm named *Double*

Propagation was applied to expand a given seed set. Zhang et al. (2010) improved Qiu's work by adding more feasible syntactic patterns, and the HITS algorithm (Kleinberg, 1999) was employed to rank candidates. Moghaddam and Ester (2010) extracted product features by automatical opinion pattern mining. Zhuang et al. (2006) used various syntactic templates from an annotated movie corpus and applied them to supervised movie feature extraction. Wu et al. (2009) proposed a phrase level dependency parsing for mining aspects and features of products.

As discussed in the first section, syntactic patterns often suffer from data sparsity. Furthermore, most pattern-based methods rely on term frequency, which have the limitation of finding infrequent but important product features. A recent research (Xu et al., 2013) extracted infrequent product features by a semi-supervised classifier, which used word-syntactic pattern co-occurrence statistics as features for the classifier. However, this kind of feature is still sparse for infrequent candidates. Our method adopts a semantic word representation model, which can train dense features unsupervisedly on a very large corpus. Thus, the data sparsity problem can be alleviated.

3 The Proposed Method

We propose a semantics-based bootstrapping method for product feature mining. Firstly, some product feature seeds are automatically extracted. Then, a semantic similarity graph is created to capture lexical semantic clue, and a Convolutional Neural Network (CNN) (Collobert et al., 2011) is trained in each bootstrapping iteration to encode contextual semantic clue. Finally we use Label Propagation to find some reliable new seeds for the training of the next bootstrapping iteration.

3.1 Automatic Seed Generation

The seed set consists of positive labeled examples (i.e. product features) and negative labeled examples (i.e. noise terms). Intuitively, popular product features are frequently mentioned in reviews, so they can be extracted by simply mining frequently occurring nouns (Hu and Liu, 2004). However, this strategy will also find many noise terms (e.g., commonly used nouns like *thing*, *one*, etc.). To produce high quality seeds, we employ a Domain Relevance Measure (DRM) (Jiang and Tan, 2010), which combines term frequency with a domain-specific measuring metric called Likelihood Ratio Test (LRT) (Dunning, 1993). Let $\lambda(t)$ denotes the LRT score of a product feature candidate t,

$$\lambda(t) = \frac{p^{k_1}(1-p)^{n_1-k_1}p^{k_2}(1-p)^{n_2-k_2}}{p_1^{k_1}(1-p_1)^{n_1-k_1}p_2^{k_2}(1-p_2)^{n_2-k_2}} \quad (1)$$

where k_1 and k_2 are the frequencies of t in the review corpus $\mathcal{R}$ and a background corpus[1] $\mathcal{B}$, n_1 and n_2 are the total number of terms in $\mathcal{R}$ and $\mathcal{B}$, $p = (k_1 + k_2)/(n_1 + n_2)$, $p_1 = k_1/n_1$ and $p_2 = k_2/n_2$. Then a modified DRM[2] is proposed,

$$DRM(t) = \frac{tf(t)}{max[tf(\cdot)]} \times \frac{1}{\log df(t)} \times \frac{|\log\lambda(t)| - min|\log\lambda(\cdot)|}{max|\log\lambda(\cdot)| - min|\log\lambda(\cdot)|} \quad (2)$$

where $tf(t)$ is the frequency of t in $\mathcal{R}$ and $df(t)$ is the frequency of t in $\mathcal{B}$.

All nouns in $\mathcal{R}$ are ranked by $DRM(t)$ in descent order, where top N nouns are taken as the positive example set V_s^+. On the other hand, Xu et al. (2013) show that a set of general nouns seldom appear to be product features. Therefore, we employ their General Noun Corpus to create the negative example set V_s^-, where N most frequent terms are selected. Besides, it is guaranteed that $V_s^+ \cap V_s^- = \emptyset$, i.e., conflicting terms are taken as negative examples.

3.2 Capturing Lexical Semantic Clue in a Semantic Similarity Graph

To capture lexical semantic clue, each word is first converted into *word embedding*, which is a continuous vector with each dimension's value corresponds to a semantic or grammatical interpretation (Turian et al., 2010). Learning large-scale word embeddings is very time-consuming (Collobert et al., 2011), we thus employ a faster method named Skip-gram model (Mikolov et al., 2013).

3.2.1 Learning Word Embedding for Semantic Representation

Given a sequence of training words $W = \{w_1, w_2, ..., w_m\}$, the goal of the Skip-gram model is to learn a continuous vector space $EB = \{e_1, e_2, ..., e_m\}$, where e_i is the word embedding of w_i. The training objective is to maximize the

[1] Google-n-Gram (http://books.google.com/ngrams) is used as the background corpus.

[2] The $df(t)$ part of the original DRM is slightly modified because we want a $tf \times idf$-like scheme (Liu et al., 2012).

average log probability of using word w_t to predict a surrounding word w_{t+j},

$$\hat{EB} = \underset{e_t \in EB}{\text{argmax}} \frac{1}{m} \sum_{t=1}^{m} \sum_{-c \le j \le c, j \neq 0} \log p(w_{t+j}|w_t; e_t) \quad (3)$$

where c is the size of the training window. Basically, $p(w_{t+j}|w_t; e_t)$ is defined as,

$$p(w_{t+j}|w_t; e_t) = \frac{exp(e'^T_{t+j} e_t)}{\sum_{w=1}^{m} exp(e'^T_w e_t)} \quad (4)$$

where e'_i is an additional training vector associated with e_i. This basic formulation is impractical because it is proportional to m. A hierarchical softmax approximation can be applied to reduce the computational cost to $log_2(m)$, see (Morin and Bengio, 2005) for details.

To alleviate the data sparsity problem, EB is first trained on a very large corpus[3] (denoted by $\mathcal{C}$), and then fine-tuned on the target review corpus $\mathcal{R}$. Particularly, for phrasal product features, a statistic-based method in (Zhu et al., 2009) is used to detect noun phrases in $\mathcal{R}$. Then, an Unfolding Recursive Autoencoder (Socher et al., 2011) is trained on $\mathcal{C}$ to obtain embedding vectors for noun phrases. In this way, semantics of infrequent terms in $\mathcal{R}$ can be well captured. Finally, the phrase-based Skip-gram model in (Mikolov et al., 2013) is applied on $\mathcal{R}$.

3.2.2 Building the Semantic Similarity Graph

Lexical semantic clue is captured by measuring semantic similarity between terms. The underlying motivation is that if we have known some product feature seeds, then terms that are more semantically similar to these seeds are more likely to be product features. For example, if *screen* is known to be a product feature of *mp3*, and *lcd* is of high semantic similarity with *screen*, we can infer that *lcd* is also a product feature. Analogously, terms that are semantically similar to negative labeled seeds are not product features.

Word embedding naturally meets the demand above: words that are more semantically similar to each other are located closer in the embedding space (Collobert et al., 2011). Therefore, we can use cosine distance between two embedding vectors as the semantic distance measuring metric. Thus, our method does not rely on term frequency to rank candidates. This could potentially improve the ability of mining infrequent product features.

Formally, we create a semantic similarity graph $G = (V, E, W)$, where $V = \{V_s \cup V_c\}$ is the vertex set, which contains the labeled seed set V_s and the unlabeled candidate set V_c; E is the edge set which connects every vertex pair (u, v), where $u, v \in V$; $W = \{w_{uv} : \cos(EB_u, EB_v)\}$ is a function which associates a weight to each edge.

3.3 Encoding Contextual Semantic Clue Using Convolutional Neural Network

The CNN is trained on each occurrence of seeds that is found in review texts. Then for a candidate term t, the CNN classifies all of its occurrences. Since seed terms tend to have high frequency in review texts, only a few seeds will be enough to provide plenty of occurrences for the training.

3.3.1 The architecture of the Convolutional Neural Network

The architecture of the Convolutional Neural Network is shown in Figure 2. For a product feature candidate t in sentence s, every consecutive subsequence q_i of s that containing t with a window of length l is fed to the CNN. For example, as in Figure 2, if $t = \{screen\}$, and $l = 3$, there are three inputs: $q_1 = [the, ipod, \textbf{screen}]$, $q_2 = [ipod, \textbf{screen}, is]$, $q_3 = [\textbf{screen}, is, impressive]$. Partially, t is replaced by a token "*PF*" to remove its lexicon influence[4].

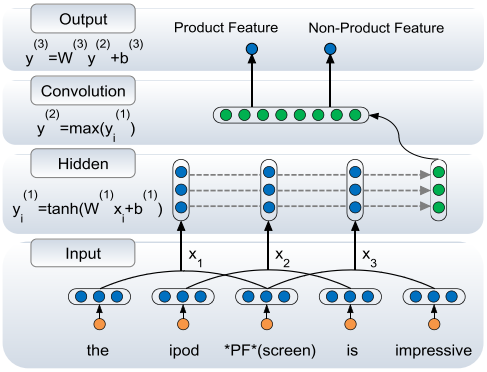


Figure 2: The architecture of the Convolutional Neural Network.

To get the output score, q_i is first converted into a concatenated vector $x_i = [e_1; e_2; ...; e_l]$, where e_j is the word embedding of the *j*-th word. In this way, the CNN serves as a *soft* pattern miner:

[3]Wikipedia(http://www.wikipedia.org) is used in practice.

[4]Otherwise, the CNN will quickly get overfitting on *t*, because very few seed lexicons are used for the training.

since words that have similar semantics have similar low-dimension embedding vectors, the CNN is less sensitive to lexicon change. The network is computed by,

$$y_i^{(1)} = tanh(W^{(1)}x_i + b^{(1)}) \quad (5)$$

$$y^{(2)} = max(y_i^{(1)}) \quad (6)$$

$$y^{(3)} = W^{(3)}y^{(2)} + b^{(3)} \quad (7)$$

where $y^{(i)}$ is the output score of the *i*-th layer, and $b^{(i)}$ is the bias of the *i*-th layer; $W^{(1)} \in \mathbb{R}^{h\times(nl)}$ and $W^{(3)} \in \mathbb{R}^{2\times h}$ are parameter matrixes, where n is the dimension of word embedding, and h is the size of nodes in the hidden layer.

In conventional neural models, the candidate term t is placed in the center of the window. However, from Example 2, when $l = 5$, we can see that the best windows should be the bracketed texts (Because, intuitively, the windows should contain *mp3*, which is a strong evidence for finding the product feature), where $t = \{screen\}$ is at the boundary. Therefore, we use Equ. 6 to formulate a max-convolutional layer, which is aimed to enable the CNN to find more evidences in contexts than conventional neural models.

Example 2:
(a) *The [**screen** of this mp3 is] great.*
(b) *This [mp3 has a great **screen**].*

3.3.2 Training

Let $\theta = \{EB, W^{(\cdot)}, b^{(\cdot)}\}$ denotes all the trainable parameters. The softmax function is used to convert the output score of the CNN to a probability,

$$p(t|X;\theta) = \frac{exp(y^{(3)})}{\sum_{j=1}^{|C|} exp(y_j^{(3)})} \quad (8)$$

where X is the input set for term t, and $C = \{0, 1\}$ is the label set representing product feature and non-product feature, respectively.

To train the CNN, we first use V_s to collect each occurrence of the seeds in $\mathcal{R}$ to form a training set T_s. Then, the training criterion is to minimize cross-entropy over T_s,

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{|T_s|} -\log \delta_i p(t_i|X_i;\theta) \quad (9)$$

where δ_i is the binomial target label distribution for one entry. Backpropagation algorithm with mini-batch stochastic gradient descent is used to solve this optimization problem. In addition, some useful tricks can be applied during the training. The weight matrixes $W^{(\cdot)}$ are initialized by *normalized initialization* (Glorot and Bengio, 2010). $W^{(1)}$ is pre-trained by an autoencoder (Hinton, 1989) to capture semantic compositionality. To speed up the learning, a momentum method is applied (Sutskever et al., 2013).

3.4 Combining Lexical and Contextual Semantic Clues by Label Propagation

We propose a Label Propagation algorithm to combine both semantic clues in a unified process. Each term $t \in V$ is assumed to have a label distribution $L_t = (p_t^+, p_t^-)$, where p_t^+ denotes the probability of the candidate being a product feature, and on the contrary, $p_t^- = 1 - p_t^+$. The classified results of the CNN which encode contextual semantic clue serve as the prior knowledge,

$$I_t = \begin{cases} (1,0), & if\ t \in V_s^+ \\ (0,1), & if\ t \in V_s^- \\ (r_t^+, r_t^-), & if\ t \in V_c \end{cases} \quad (10)$$

where (r_t^+, r_t^-) is estimated by,

$$r_t^+ = \frac{count^+(t)}{count^+(t) + count^-(t)} \quad (11)$$

where $count^+(t)$ is the number of occurrences of term t that are classified as positive by the CNN, and $count^-(t)$ represents the negative count.

Label Propagation is applied to propagate the prior knowledge distribution I to the product feature distribution L via semantic similarity graph G, so that a product feature candidate is determined by exploring its semantic relations to all of the seeds and other candidates globally. We propose an adapted version on the random walking view of the Adsorption algorithm (Baluja et al., 2008) by updating the following formula until L converges,

$$L^{i+1} = (1-\alpha)\mathbf{M}^T L^i + \alpha \mathbf{D} I \quad (12)$$

where $\mathbf{M}$ is the semantic transition matrix built from G; $\mathbf{D} = Diag[\log tf(t)]$ is a diagonal matrix of log frequencies, which is designed to assign higher "confidence" scores to more frequent seeds; and α is a balancing parameter. Particularly, when $\alpha = 0$, we can set the prior knowledge I without V_c to L^0 so that only lexical semantic clue is used; otherwise if $\alpha = 1$, only contextual semantic clue is used.

3.5 The Bootstrapping Framework

We summarize the bootstrapping framework of the proposed method in Algorithm 1. During bootstrapping, the CNN is enhanced by Label Propagation which finds more labeled examples for training, and then the performance of Label Propagation is also improved because the CNN outputs a more accurate prior distribution. After running for several iterations, the algorithm gets enough seeds, and a final Label Propagation is conducted to produce the results.

Algorithm 1: Bootstrapping using semantic clues

Input: The review corpus $\mathcal{R}$, a large corpus $\mathcal{C}$
Output: The mined product feature list P
Initialization: Train word embedding set EB first on $\mathcal{C}$, and then on $\mathcal{R}$

Step 1: Generate product feature seeds V_s (Section 3.1)

Step 2: Build semantic similarity graph G (Section 3.2)

while *iter* $<$ *MAX_ITER* **do**

  Step 3: Use V_s to collect occurrence set T_s from $\mathcal{R}$ for training

  Step 4: Train a CNN $\mathcal{N}$ on T_s (Section 3.3)
  Apply mini-batch SGD on Equ. 9;

  Step 5: Run Label Propagation (Section 3.4)
  Classify candidates using $\mathcal{N}$ to setup I;
  $L^0 \leftarrow I$;
  repeat
    $L^{i+1} \leftarrow (1-\alpha)\mathbf{M}^T L^i + \alpha \mathbf{D} I$;
  until $||L^{i+1} - L^i||^2 < \varepsilon$;

  Step 6: Expand product feature seeds
  Move top T terms from V_c to V_s;

  iter++

end

Step 7: Run Label Propagation for a final result L_f
Rank terms by L_f^+ to get P, where $L_f^+ > L_f^-$;

4 Experiments

4.1 Datasets and Evaluation Metrics

Datasets: We select two real world datasets to evaluate the proposed method. The first one is a benchmark dataset in Wang et al. (2011), which contains English review sets on two domains (*MP3* and *Hotel*)[5]. The second dataset is proposed by Chinese Opinion Analysis Evaluation 2008 (COAE 2008)[6], where two review sets (*Camera* and *Car*) are selected. Xu et al. (2013) had manually annotated product features on these four domains, so we directly employ their annotation as the gold standard. The detailed information can be found in their original paper.

[5]http://timan.cs.uiuc.edu/downloads.html
[6]http://ir-china.org.cn/coae2008.html

Evaluation Metrics: We evaluate the proposed method in terms of precision(P), recall(R) and F-measure(F). The English results are evaluated by *exact* string match. And for Chinese results, we use an *overlap* matching metric, because determining the exact boundaries is hard even for human (Wiebe et al., 2005).

4.2 Experimental Settings

For English corpora, the pre-processing are the same as that in (Qiu et al., 2009), and for Chinese corpora, the Stanford Word Segmenter (Chang et al., 2008) is used to perform word segmentation. We select three state-of-the-art syntax-based methods to be compared with our method:

DP uses a bootstrapping algorithm named as *Double Propagation* (Qiu et al., 2009), which is a conventional syntax-based method.

DP-HITS is an enhanced version of *DP* proposed by Zhang et al. (2010), which ranks product feature candidates by

$$s(t) = \log tf(t) * importance(t) \qquad (13)$$

where $importance(t)$ is estimated by the HITS algorithm (Kleinberg, 1999).

SGW is the Sentiment Graph Walking algorithm proposed in (Xu et al., 2013), which first extracts syntactic patterns and then uses random walking to rank candidates. Afterwards, word-syntactic pattern co-occurrence statistic is used as feature for a semi-supervised classifier TSVM (Joachims, 1999) to further refine the results. This two-stage method is denoted as ***SGW-TSVM***.

LEX only uses lexical semantic clue. Label Propagation is applied alone in a self-training manner. The dimension of word embedding $n = 100$, the convergence threshold $\varepsilon = 10^{-7}$, and the number of expanded seeds $T = 40$. The size of the seed set N is 40. To output product features, it ranks candidates in descent order by using the positive score $L_f^+(t)$.

CONT only uses contextual semantic clue, which only contains the CNN. The window size l is 5. The CNN is trained with a mini-batch size of 50. The hidden layer size $h = 250$. Finally, $importance(t)$ in Equ. 13 is replaced with r_t^+ in Equ. 11 to rank candidates.

LEX&CONT leverages both semantic clues.

Method	MP3			Hotel			Camera			Car			Avg.
	P	R	F	P	R	F	P	R	F	P	R	F	F
DP	0.66	0.57	0.61	0.66	0.60	0.63	0.71	0.70	0.70	0.72	0.65	0.68	0.66
DP-HITS	0.65	0.62	0.63	0.64	0.66	0.65	0.71	0.78	0.74	0.69	0.68	0.68	0.68
SGW	0.62	0.68	0.65	0.63	0.71	0.67	0.69	0.80	0.74	0.66	0.71	0.68	0.69
LEX	0.64	0.74	0.69	0.65	0.75	0.70	0.69	0.84	0.76	0.68	0.78	0.73	0.72
CONT	0.68	0.65	0.66	0.69	0.68	0.68	0.74	0.77	0.75	0.74	0.70	0.72	0.71
SGW-TSVM	0.73	0.71	0.72	0.75	0.73	0.74	0.78	0.81	0.79	0.76	0.73	0.74	0.75
LEX&CONT	0.74	0.75	0.74	0.75	0.77	0.76	0.80	0.84	0.82	0.79	0.79	0.79	0.78

Table 1: Experimental results of product feature mining. The precision or recall of *CONT* is the average performance over five runs with different random initialization of parameters of the CNN. Avg. stands for the average score.

4.3 The Semantics-based Methods vs. State-of-the-art Syntax-based Methods

The experimental results are shown in Table 1, from which we have the following observations:

(i) Our method achieves the best performance among all of the compared methods. We also equally split the dataset into five subsets, and perform one-tailed t-test ($p \leq 0.05$), which shows that the proposed semantics-based method (*LEX&CONT*) significantly outperforms the three syntax-based strong competitors (*DP*, *DP-HITS* and *SGW-TSVM*).

(ii) *LEX&CONT* which leverages both lexical and contextual semantic clues outperforms approaches that only use one kind of semantic clue (*LEX* and *CONT*), showing that the combination of the semantic clues is helpful.

(iii) Our methods which use only one kind of semantic clue (*LEX* and *CONT*) outperform syntax-based methods (*DP*, *DP-HITS* and *SGW*). Comparing *DP-HITS* with *LEX* and *CONT*, the difference between them is that *DP-HITS* uses a syntax-pattern-based algorithm to estimate $importance(t)$ in Equ. 13, while our methods use lexical or contextual semantic clue instead. We believe the reason that *LEX* or *CONT* is better is that syntactic patterns only use discrete and local information. In contrast, *CONT* exploits latent semantics of each word in context, and *LEX* takes advantage of word embedding, which is induced from global word co-occurrence statistic. Furthermore, comparing *SGW* and *LEX*, both methods are base on random surfer model, but *LEX* gets better results than *SGW*. Therefore, the word-word semantic similarity relation used in *LEX* is more reliable than the word-syntactic pattern relation used in *SGW*.

(iv) *LEX&CONT* achieves the highest recall among all of the evaluated methods. Since *DP* and *DP-HITS* rely on frequency for ranking product features, infrequent candidates are ranked low in their extracted list. As for *SGW-TSVM*, the features they used for the TSVM suffer from the data sparsity problem for infrequent terms. In contrast, *LEX&CONT* is frequency-independent to the review corpus. Further discussions on this observation are given in the next section.

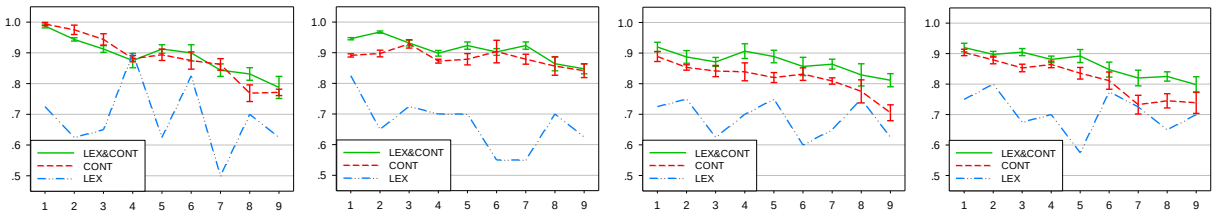
4.4 The Results on Extracting Infrequent Product Features

We conservatively regard 30% product features with the highest frequencies in $\mathcal{R}$ as *frequent features*, so the remaining terms in the gold standard are *infrequent features*. In product feature mining task, frequent features are relatively easy to find. Table 2 shows the recall of all the four approaches for mining frequent product features. We can see that the performance are very close among different methods. Therefore, the recall mainly depends on mining the infrequent features.

Method	MP3	Hotel	Camera	Car
DP	0.89	0.92	0.86	0.84
DP-HITS	0.89	0.91	0.86	0.85
SGW-TSVM	0.87	0.92	0.88	0.87
LEX&CONT	0.89	0.91	0.89	0.87

Table 2: The recall of frequent product features.

Figure 3 gives the recall of infrequent product features, where *LEX&CONT* achieves the best performance. So our method is less influenced by term frequency. Furthermore, *LEX* gets better recall than *CONT* and all syntax-based methods, which indicates that lexical semantic clue does aid to mine more infrequent features as expected.



(a) MP3 (b) Hotel (c) Camera (d) Car

Figure 4: Accuracy (y-axis) of product feature seed expansion at each bootstrapping iteration (x-axis). The error bar shows the standard deviation over five runs.

Method	MP3			Hotel			Camera			Car		
	P	R	F	P	R	F	P	R	F	P	R	F
FW-5	0.62	0.63	0.62	0.64	0.64	0.64	0.68	0.73	0.70	0.67	0.66	0.66
FW-9	0.64	0.65	0.64	0.66	0.68	0.67	0.70	0.76	0.73	0.71	0.70	0.70
CONT	0.68	0.65	0.66	0.69	0.68	0.68	0.74	0.77	0.75	0.74	0.70	0.72

Table 3: The results of convolutional method vs. the results of non-convolutional methods.

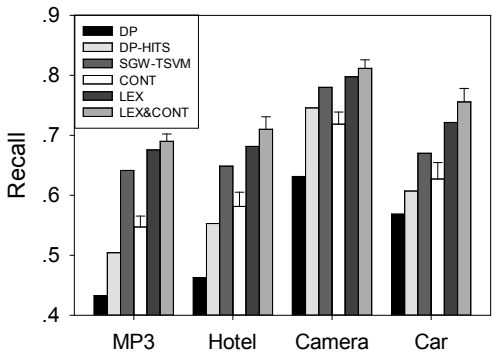


Figure 3: The recall of infrequent features. The error bar shows the standard deviation over five different runs.

4.5 Lexical Semantic Clue vs. Contextual Semantic Clue

This section studies the effects of lexical semantic clue and contextual semantic clue during seed expansion (Step 6 in Algorithm 1), which is controlled by α. When $\alpha = 1$, we get the *CONT*; and if α is set 0, we get the *LEX*. To take into account the correctly expanded terms for both positive and negative seeds, we use Accuracy as the evaluation metric,

$$Accuracy = \frac{\#TP + \#TN}{\#\ Extracted\ Seeds}$$

where TP denotes the *true positive* seeds, and TN denotes the *true negative* seeds.

Figure 4 shows the performance of seed expansion during bootstrapping, in which the accuracy is computed on 40 seeds (20 being positive and 20 being negative) expanded in each iteration. We can see that the accuracies of *CONT* and *LEX&CONT* retain at a high level, which shows that they can find reliable new product feature seeds. However, the performance of *LEX* oscillates sharply and it is very low for some points, which indicates that using lexical semantic clue alone is infeasible. On another hand, comparing *CONT* with *LEX* in Table 1, we can see that *LEX* performs generally better than *CONT*. Although *LEX* is not so accurate as *CONT* during seed expansion, its final performance surpasses *CONT*. Consequently, we can draw conclusion that *CONT* is more suitable for the seed expansion, and *LEX* is more robust for the final result production.

To combine advantages of the two kinds of semantic clues, we set $\alpha = 0.7$ in Step 5 of Algorithm 1, so that contextual semantic clue plays a key role to find new seeds accurately. For Step 7, we set $\alpha = 0.3$. Thus, lexical semantic clue is emphasized for producing the final results.

4.6 The Effect of Convolutional Layer

Two non-convolutional variations of the proposed method are used to be compared with the convolutional method in *CONT*. ***FW-5*** uses a traditional neural network with a fixed window size of 5 to replace the CNN in *CONT*, and the candidate term to be classified is placed in the center of the window. Similarly, ***FW-9*** uses a fixed window size of 9. Note that *CONT* uses a 5-term dynamic window containing the candidate term, so the exploited number of words in the context is equivalent to *FW-9*.

Table 3 shows the experimental results. We can see that the performance of *FW-5* is much worse than *CONT*. The reason is that *FW-5* only exploits half of the context as that of *CONT*, which is not sufficient enough. Meanwhile, although *FW-9* exploits equivalent range of context as that of *CONT*, it gets lower precisions. It is because *FW-9* has approximately two times parameters in the parameter matrix $W^{(1)}$ than that in Equ. 5 of *CONT*, which makes it more difficult to be trained with the same amount of data. Also, lengths of many sentences in the review corpora are shorter than 9. Therefore, the convolutional approach in *CONT* is the most effective way among these settings.

4.7 Parameter Study

We investigate two key parameters of the proposed method: the initial number of seeds N, and the size of the window l used by the CNN.

Figure 5 shows the performance under different N, where the F-Measure saturates when N equates to 40 and beyond. Hence, very few seeds are needed for starting our algorithm.

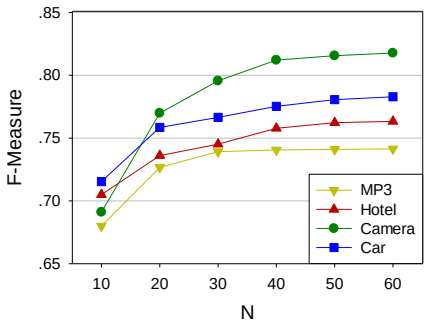


Figure 5: F-Measure vs. N for the final results.

Figure 6 shows F-Measure under different window size l. We can see that the performance is improved little when l is larger than 5. Therefore, $l = 5$ is a proper window size for these datasets.

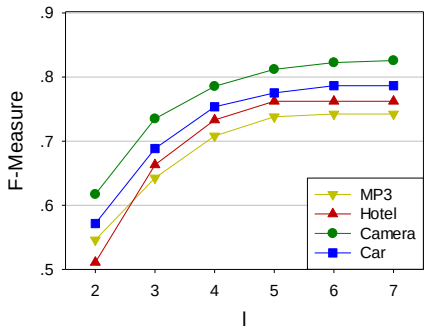


Figure 6: F-Measure vs. l for the final results.

5 Conclusion and Future Work

This paper proposes a product feature mining method by leveraging contextual and lexical semantic clues. A semantic similarity graph is built to capture lexical semantic clue, and a convolutional neural network is used to encode contextual semantic clue. Then, a Label Propagation algorithm is applied to combine both semantic clues. Experimental results prove the effectiveness of the proposed method, which not only mines product features more accurately than conventional syntax-based method, but also extracts more infrequent product features.

In future work, we plan to extend the proposed method to jointly mine product features along with customers' opinions on them. The learnt semantic representations of words may also be utilized to predict fine-grained sentiment distributions over product features.

Acknowledgement

This work was sponsored by the National Basic Research Program of China (No. 2012CB316300), the National Natural Science Foundation of China (No. 61272332 and No. 61202329), the National High Technology Development 863 Program of China (No. 2012AA011102), and CCF-Tencent Open Research Fund. This work was also supported in part by Noahs Ark Lab of Huawei Tech. Ltm.

References

Shumeet Baluja, Rohan Seth, D. Sivakumar, Yushi Jing, Jay Yagnik, Shankar Kumar, Deepak Ravichandran, and Mohamed Aly. 2008. Video suggestion and discovery for youtube: Taking random walks through the view graph. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 895–904, New York, NY, USA. ACM.

Pi-Chuan Chang, Michel Galley, and Christopher D. Manning. 2008. Optimizing chinese word segmentation for machine translation performance. In *Proceedings of the Third Workshop on Statistical Machine Translation*, StatMT '08, pages 224–232.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed

dependency parses from phrase structure parses. In *Proceedings of the IEEE / ACL'06 Workshop on Spoken Language Technology*.

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Comput. Linguist.*, 19(1):61–74, March.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics.*

Geoffrey E. Hinton. 1989. Connectionist learning procedures. *Artificial Intelligence*, 40(1C3):185 – 234.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA. ACM.

Xing Jiang and Ah-Hwee Tan. 2010. Crctol: A semantic-based domain ontology learning system. *Journal of the American Society for Information Science and Technology*, 61(1):150–168.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 151–160, Stroudsburg, PA, USA. Association for Computational Linguistics.

Thorsten Joachims. 1999. Transductive inference for text classification using support vector machines. In *Proceedings of the 16th International Conference on Machine Learning*, pages 200–209.

Jon M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, September.

Fangtao Li, Sinno Jialin Pan, Ou Jin, Qiang Yang, and Xiaoyan Zhu. 2012. Cross-domain co-extraction of sentiment and topic lexicons. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 410–419, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kang Liu, Liheng Xu, and Jun Zhao. 2012. Opinion target extraction using word-based translation model. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1346–1356, Jeju Island, Korea, July. Association for Computational Linguistics.

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

Samaneh Moghaddam and Martin Ester. 2010. Opinion digger: An unsupervised opinion miner from unstructured product reviews. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 1825–1828, New York, NY, USA. ACM.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of the international workshop on artificial intelligence and statistics*, AISTATS05, pages 246–252.

Ana-Maria Popescu and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 339–346.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2009. Expanding domain sentiment lexicon through double propagation. In *Proceedings of the 21st international jont conference on Artifical intelligence*, IJCAI'09, pages 1199–1204.

Richard Socher, Eric H Huang, Jeffrey Pennington, Andrew Y Ng, and Christopher D Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *NIPS'2011*, volume 24, pages 801–809.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 30 th International Conference on Machine Learning*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hongning Wang, Yue Lu, and ChengXiang Zhai. 2011. Latent aspect rating analysis without aspect keyword supervision. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 618–626, New York, NY, USA. ACM.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39(2-3):165–210.

Yuanbin Wu, Qi Zhang, Xuanjing Huang, and Lide Wu. 2009. Phrase dependency parsing for opinion mining. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, EMNLP '09, pages 1533–1541, Stroudsburg, PA, USA. Association for Computational Linguistics.

Liheng Xu, Kang Liu, Siwei Lai, Yubo Chen, and Jun Zhao. 2013. Mining opinion words and opinion targets in a two-stage framework. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1764–1773, Sofia, Bulgaria, August. Association for Computational Linguistics.

Lei Zhang, Bing Liu, Suk Hwan Lim, and Eamonn O'Brien-Strain. 2010. Extracting and ranking product features in opinion documents. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 1462–1470, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jingbo Zhu, Huizhen Wang, Benjamin K. Tsou, and Muhua Zhu. 2009. Multi-aspect opinion polling from textual reviews. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 1799–1802, New York, NY, USA. ACM.

Li Zhuang, Feng Jing, and Xiao-Yan Zhu. 2006. Movie review mining and summarization. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, CIKM '06, pages 43–50, New York, NY, USA. ACM.

Aspect Extraction with Automated Prior Knowledge Learning

Zhiyuan Chen Arjun Mukherjee Bing Liu
Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60607, USA
{czyuanacm,arjun4787}@gmail.com,liub@cs.uic.edu

Abstract

Aspect extraction is an important task in sentiment analysis. Topic modeling is a popular method for the task. However, unsupervised topic models often generate incoherent aspects. To address the issue, several knowledge-based models have been proposed to incorporate prior knowledge provided by the user to guide modeling. In this paper, we take a major step forward and show that in the big data era, without any user input, it is possible to learn prior knowledge automatically from a large amount of review data available on the Web. Such knowledge can then be used by a topic model to discover more coherent aspects. There are two key challenges: (1) learning quality knowledge from reviews of diverse domains, and (2) making the model fault-tolerant to handle possibly wrong knowledge. A novel approach is proposed to solve these problems. Experimental results using reviews from 36 domains show that the proposed approach achieves significant improvements over state-of-the-art baselines.

1 Introduction

Aspect extraction aims to extract target entities and their aspects (or attributes) that people have expressed opinions upon (Hu and Liu, 2004, Liu, 2012). For example, in "*The voice is not clear*," the aspect term is "*voice*." Aspect extraction has two subtasks: *aspect term extraction* and *aspect term resolution*. Aspect term resolution groups extracted synonymous aspect terms together. For example, "voice" and "sound" should be grouped together as they refer to the same aspect of phones.

Recently, topic models have been extensively applied to aspect extraction because they can perform both subtasks at the same time while other existing methods all need two separate steps (see Section 2). Traditional topic models such as LDA (Blei et al., 2003) and pLSA (Hofmann, 1999) are unsupervised methods for extracting latent topics in text documents. Topics are aspects in our task. Each aspect (or topic) is a distribution over (aspect) terms. However, researchers have shown that fully unsupervised models often produce incoherent topics because the objective functions of topic models do not always correlate well with human judgments (Chang et al., 2009).

To tackle the problem, several semi-supervised topic models, also called knowledge-based topic models, have been proposed. DF-LDA (Andrzejewski et al., 2009) can incorporate two forms of prior knowledge from the user: *must-links* and *cannot-links*. A must-link implies that two terms (or words) should belong to the same topic whereas a cannot-link indicates that two terms should not be in the same topic. In a similar but more generic vein, *must-sets* and *cannot-sets* are used in MC-LDA (Chen et al., 2013b). Other related works include (Andrzejewski et al., 2011, Chen et al., 2013a, Chen et al., 2013c, Mukherjee and Liu, 2012, Hu et al., 2011, Jagarlamudi et al., 2012, Lu et al., 2011, Petterson et al., 2010). They all allow prior knowledge to be specified by the user to guide the modeling process.

In this paper, we take a major step further. We mine the prior knowledge directly from a large amount of relevant data without any user intervention, and thus make this approach fully automatic. We hypothesize that it is possible to learn quality prior knowledge from the big data (of reviews) available on the Web. The intuition is that although every domain is different, there is a decent amount of aspect overlapping across domains. For example, every product domain has the aspect/topic of "*price*," most electronic products share the aspect "*battery*" and some also share "*screen*." Thus, the shared aspect knowl-

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 347–358,
Baltimore, Maryland, USA, June 23-25 2014.

edge mined from a set of domains can potentially help improve aspect extraction in each of these domains, as well as in new domains. Our proposed method aims to achieve this objective. There are two major challenges: (1) learning quality knowledge from a large number of domains, and (2) making the extraction model fault-tolerant, i.e., capable of handling possibly incorrect learned knowledge. We briefly introduce the proposed method below, which consists of two steps.

Learning quality knowledge: Clearly, learned knowledge from only a single domain can be erroneous. However, if the learned knowledge is shared by multiple domains, the knowledge is more likely to be of high quality. We thus propose to first use LDA to learn topics/aspects from each individual domain and then discover the shared aspects (or topics) and aspect terms among a subset of domains. These shared aspects and aspect terms are more likely to be of good quality. They can serve as the prior knowledge to guide a model to extract aspects. A piece of knowledge is a set of semantically coherent (aspect) terms which are likely to belong to the same topic or aspect, i.e., similar to a must-link, but mined automatically.

Extraction guided by learned knowledge: For reliable aspect extraction using the learned prior knowledge, we must account for possible errors in the knowledge. In particular, a piece of automatically learned knowledge may be wrong or domain specific (i.e., the words in the knowledge are semantically coherent in some domains but not in others). To leverage such knowledge, the system must detect those inappropriate pieces of knowledge. We propose a method to solve this problem, which also results in a new topic model, called AKL (*Automated Knowledge LDA*), whose inference can exploit the automatically learned prior knowledge and handle the issues of incorrect knowledge to produce superior aspects.

In summary, this paper makes the following contributions:

1. It proposes to exploit the big data to learn prior knowledge and leverage the knowledge in topic models to extract more coherent aspects. The process is fully automatic. To the best of our knowledge, none of the existing models for aspect extraction is able to achieve this.
2. It proposes an effective method to learn quality knowledge from raw topics produced using review corpora from many different domains.
3. It proposes a new inference mechanism for topic modeling, which can handle incorrect knowledge in aspect extraction.

2 Related Work

Aspect extraction has been studied by many researchers in sentiment analysis (Liu, 2012, Pang and Lee, 2008), e.g., using supervised sequence labeling or classification (Choi and Cardie, 2010, Jakob and Gurevych, 2010, Kobayashi et al., 2007, Li et al., 2010, Yang and Cardie, 2013) and using word frequency and syntactic patterns (Hu and Liu, 2004, Ku et al., 2006, Liu et al., 2013, Popescu and Etzioni, 2005, Qiu et al., 2011, Somasundaran and Wiebe, 2009, Wu et al., 2009, Xu et al., 2013, Yu et al., 2011, Zhao et al., 2012, Zhou et al., 2013, Zhuang et al., 2006). However, these works only perform extraction but not aspect term grouping or resolution. Separate aspect term grouping has been done in (Carenini et al., 2005, Guo et al., 2009, Zhai et al., 2011). They assume that aspect terms have been extracted beforehand.

To extract and group aspects simultaneously, topic models have been applied by researchers (Branavan et al., 2008, Brody and Elhadad, 2010, Chen et al., 2013b, Fang and Huang, 2012, He et al., 2011, Jo and Oh, 2011, Kim et al., 2013, Lazaridou et al., 2013, Li et al., 2011, Lin and He, 2009, Lu et al., 2009, Lu et al., 2012, Lu and Zhai, 2008, Mei et al., 2007, Moghaddam and Ester, 2013, Mukherjee and Liu, 2012, Sauper and Barzilay, 2013, Titov and McDonald, 2008, Wang et al., 2010, Zhao et al., 2010). Our proposed AKL model belongs to the class of knowledge-based topic models. Besides the knowledge-based topic models discussed in Section 1, document labels are incorporated as implicit knowledge in (Blei and McAuliffe, 2007, Ramage et al., 2009). Geographical region knowledge has also been considered in topic models (Eisenstein et al., 2010). All of these models assume that the prior knowledge is correct. GK-LDA (Chen et al., 2013a) is the only knowledge-based topic model that deals with wrong lexical knowledge to some extent. As we will see in Section 6, AKL outperformed GK-LDA significantly due to AKL's more effective error handling mechanism. Furthermore, GK-LDA does not learn any prior knowledge.

Our work is also related to transfer learning to some extent. Topic models have been used to help

Input: Corpora $\boldsymbol{D_L}$ for knowledge learning
Test corpora $\boldsymbol{D_T}$

1: // STEP 1: Learning prior knowledge.
2: **for** $r = 0$ **to** R **do** // Iterate $R + 1$ times.
3: **for** each domain corpus $D_i \in \boldsymbol{D_L}$ **do**
4: **if** $r = 0$ **then**
5: $A_i \leftarrow \text{LDA}(D_i)$;
6: **else**
7: $A_i \leftarrow \text{AKL}(D_i, \boldsymbol{K})$;
8: **end if**
9: **end for**
10: $\boldsymbol{A} \leftarrow \cup_i A_i$;
11: $TC \leftarrow \text{Clustering}(\boldsymbol{A})$;
12: **for** each cluster $T_j \in TC$ **do**
13: $K_j \leftarrow \text{FPM}(T_j)$;
14: **end for**
15: $\boldsymbol{K} \leftarrow \cup_j K_j$;
16: **end for**
17: // STEP 2: Using the learned knowledge.
18: **for** each test corpus $D_i \in \boldsymbol{D_T}$ **do**
19: $A_i \leftarrow \text{AKL}(D_i, \boldsymbol{K})$;
20: **end for**

Figure 1: The proposed overall algorithm.

transfer learning (He et al., 2011, Pan and Yang, 2010, Xue et al., 2008). However, transfer learning in these papers is for traditional classification rather than topic/aspect extraction. In (Kang et al., 2012), labeled documents from source domains are transferred to the target domain to produce topic models with better fitting. However, we do not use any labeled data. In (Yang et al., 2011), a user provided parameter indicating the technicality degree of a domain was used to model the language gap between topics. In contrast, our method is fully automatic without human intervention.

3 Overall Algorithm

This section introduces the proposed overall algorithm. It consists of two main steps: *learning quality knowledge* and *using the learned knowledge*. Figure 1 gives the algorithm.

Step 1 (learning quality knowledge, Lines 1-16): The input is the review corpora $\boldsymbol{D_L}$ from multiple domains, from which the knowledge is automatically learned. Lines 3 and 5 run LDA on each review domain corpus $D_i \in \boldsymbol{D_L}$ to generate a set of aspects/topics A_i (lines 2, 4, and 6-9 will be discussed below). Line 10 unions the topics from all domains to give $\boldsymbol{A}$. Lines 11-14 cluster the topics in $\boldsymbol{A}$ into some coherent groups (or clusters) and then discover knowledge K_j from each group of topics using frequent pattern mining (FPM) (Han et al., 2007). We will detail these in Section 4. Each piece of the learned knowledge is a set of terms which are likely to belong to the same aspect.

Iterative improvement: The above process can actually run iteratively because the learned knowledge $\boldsymbol{K}$ can help the topic model learn better topics in each domain $D_i \in \boldsymbol{D_L}$, which results in better knowledge $\boldsymbol{K}$ in the next iteration. This iterative process is reflected in lines 2, 4, 6-9 and 16. We will examine the performance of the process at different iterations in Section 6.2. From the second iteration, we can use the knowledge learned from the previous iteration (lines 6-8). The learned knowledge is leveraged by the new model AKL, which is discussed below in Step 2.

Step 2 (using the learned knowledge, Lines 17-20): The proposed model AKL is employed to use the learned knowledge $\boldsymbol{K}$ to help topic modeling in test domains $\boldsymbol{D_T}$, which can be $\boldsymbol{D_L}$ or other unseen domains. The key challenge of this step is how to use the learned prior knowledge $\boldsymbol{K}$ effectively in AKL and deal with possible errors in $\boldsymbol{K}$. We will elaborate them in Section 5.

Scalability: the proposed algorithm is naturally scalable as both LDA and AKL run on each domain independently. Thus, for all domains, the algorithm can run in parallel. Only the resulting topics need to be brought together for knowledge learning (Step 1). These resulting topics used in learning are much smaller than the domain corpus as only a list of top terms from each topic are utilized due to their high reliability.

4 Learning Quality Knowledge

This section details Step 1 in the overall algorithm, which has three sub-steps: running LDA (or AKL) on each domain corpus, clustering the resulting topics, and mining frequent patterns from the topics in each cluster. Since running LDA is simple, we will not discuss it further. The proposed AKL model will be discussed in Section 5. Below we focus on the other two sub-steps.

4.1 Topic Clustering

After running LDA (or AKL) on each domain corpus, a set of topics is obtained. Each topic is a distribution over terms (or words), i.e., terms with their associated probabilities. Here, we use only the top terms with high probabilities. As discussed earlier, quality knowledge should be shared

by topics across several domains. Thus, it is natural to exploit a frequency-based approach to discover frequent set of terms as quality knowledge. However, we need to deal with two issues.

1. Generic aspects, such as *price* with aspect terms like *cost* and *pricy*, are shared by many (even all) product domains. But specific aspects such as *screen*, occur only in domains with products having them. It means that different aspects may have distinct frequencies. Thus, using a single frequency threshold in the frequency-based approach is not sufficient to extract both generic and specific aspects because the generic aspects will result in numerous spurious aspects (Han et al., 2007).
2. A term may have multiple senses in different domains. For example, *light* can mean "of little weight" or "something that makes things visible". A good knowledge base should have the capacity of handling this ambiguity.

To deal with these two issues, we propose to discover knowledge in two stages: topic clustering and frequent pattern mining (FPM).

The purpose of clustering is to group raw topics from a topic model (LDA or AKL) into clusters. Each cluster contains semantically related topics likely to indicate the same real-world aspect. We then mine knowledge from each cluster using an FPM technique. Note that the multiple senses of a term can be distinguished by the semantic meanings represented by the topics in different clusters.

For clustering, we tried k-means and k-medoids (Kaufman and Rousseeuw, 1990), and found that k-medoids performs slightly better. One possible reason is that k-means is more sensitive to outliers. In our topic clustering, each data point is a topic represented by its top terms (with their probabilities normalized). The distance between two data points is measured by symmetrised KL-Divergence.

4.2 Frequent Pattern Mining

Given topics within each cluster, this step finds sets of terms that appear together in multiple topics, i.e., shared terms among similar topics across multiple domains. Terms in such a set are likely to belong to the same aspect. To find such sets of terms within each cluster, we use *frequent pattern mining* (FPM) (Han et al., 2007), which is suited for the task. The probability of each term is ignored in FPM.

FPM is stated as follows: Given a set of transactions $\mathbb{T}$, where each transaction $t_i \in \mathbb{T}$ is a set of items from a global item set $\mathbb{I}$, i.e., $t_i \in \mathbb{I}$. In our context, t_i is the topic vector comprising the top terms of a topic (no probability attached). $\mathbb{T}$ is the collection of all topics within a cluster and $\mathbb{I}$ is the set of all terms in $\mathbb{T}$. The goal of FPM is to find all patterns that satisfy some user-specified frequency threshold (also called minimum support count), which is the minimum number of times that a pattern should appear in $\mathbb{T}$. Such patterns are called frequent patterns. In our context, a pattern is a set of terms which have appeared multiple times in the topics within a cluster. Such patterns compose our knowledge base as shown below.

4.3 Knowledge Representation

As the knowledge is extracted from each cluster individually, we represent our knowledge base as a set of clusters, where each cluster consists of a set of frequent 2-patterns mined using FPM, e.g.,

Cluster 1: {battery, life}, {battery, hour}, {battery, long}, {charge, long}

Cluster 2: {service, support}, {support, customer}, {service, customer}

Using two terms in a set is sufficient to cover the semantic relationship of the terms belonging to the same aspect. Longer patterns tend to contain more errors since some terms in a set may not belong to the same aspect as others. Such partial errors hurt performance in the downstream model.

5 AKL: Using the Learned Knowledge

We now present the proposed topic model AKL, which is able to use the automatically learned knowledge to improve aspect extraction.

5.1 Plate Notation

Differing from most topic models based on topic-term distribution, AKL incorporates a latent cluster variable $\boldsymbol{c}$ to connect topics and terms. The plate notation of AKL is shown in Figure 2. The inputs of the model are M documents, T topics and C clusters. Each document m has N_m terms. We model distribution $P(cluster|topic)$ as ψ and distribution $P(term|topic, cluster)$ as φ with Dirichlet priors β and γ respectively. $P(topic|document)$ is modeled by θ with a Dirichlet prior α. The terms in each document are assumed to be generated by first sampling a topic z, and then a cluster c given topic z, and finally

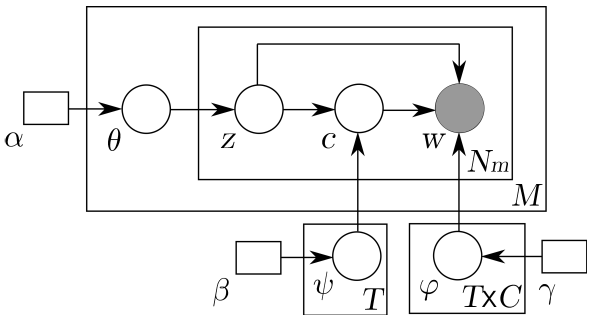


Figure 2: Plate notation for AKL.

a term w given topic z and cluster c. This plate notation of AKL and its associated generative process are similar to those of MC-LDA (Chen et al., 2013b). However, there are three key differences.

1. Our knowledge is automatically mined which may have errors (or noises), while the prior knowledge for MC-LDA is manually provided and assumed to be correct. As we will see in Section 6, using our knowledge, MC-LDA does not generate as coherent aspects as AKL.
2. Our knowledge is represented as clusters. Each cluster contains a set of frequent 2-patterns with semantically correlated terms. They are different from must-sets used in MC-LDA.
3. Most importantly, due to the use of the new form of knowledge, AKL's inference mechanism (Gibbs sampler) is entirely different from that of MC-LDA (Section 5.2), which results in superior performances (Section 6). Note that the inference mechanism and the prior knowledge cannot be reflected in the plate notation for AKL in Figure 2.

In short, our modeling contributions are (1) the capability of handling more expressive knowledge in the form of clusters, (2) a novel Gibbs sampler to deal with inappropriate knowledge.

5.2 The Gibbs Sampler

As the automatically learned prior knowledge may contain errors for a domain, AKL has to learn the usefulness of each piece of knowledge dynamically during inference. Instead of assigning weights to each piece of knowledge as a fixed prior in (Chen et al., 2013a), we propose a new Gibbs sampler, which can dynamically balance the use of prior knowledge and the information in the corpus during the Gibbs sampling iterations.

We adopt a Blocked Gibbs sampler (Rosen-Zvi et al., 2010) as it improves convergence and reduces autocorrelation when the variables (topic z and cluster c in AKL) are highly related. For each term w_i in each document, we jointly sample a topic z_i and cluster c_i (containing w_i) based on the conditional distribution in Gibbs sampler (will be detailed in Equation 4). To compute this distribution, instead of considering how well z_i matches with w_i only (as in LDA), we also consider two other factors:

1. The extent c_i corroborates w_i given the corpus. By "corroborate", we mean whether those frequent 2-patterns in c_i containing w_i are also supported by the actual information in the domain corpus to some extent (see the measure in Equation 1 below). If c_i corroborates w_i well, c_i is likely to be useful, and thus should also provide guidance in determining z_i. Otherwise, c_i may not be a suitable piece of knowledge for w_i in the domain.
2. Agreement between c_i and z_i. By agreement we mean the degree that the terms (union of all frequent 2-patterns of c_i) in cluster c_i are reflected in topic z_i. Unlike the first factor, this is a global factor as it concerns all the terms in a knowledge cluster.

For the first factor, we measure how well c_i corroborates w_i given the corpus based on co-document frequency ratio. As shown in (Mimno et al., 2011), co-document frequency is a good indicator of term correlation in a domain. Following (Mimno et al., 2011), we define a symmetric co-document frequency ratio as follows:

$$Co\text{-}Doc(w, w') = \frac{D(w, w') + 1}{(D(w) + D(w')) \times \frac{1}{2} + 1} \quad (1)$$

where (w, w') refers to each frequent 2-pattern in the knowledge cluster c_i. $D(w, w')$ is the number of documents that contain both terms w and w' and $D(w)$ is the number of documents containing w. A smoothing count of 1 is added to avoid the ratio being 0.

For the second factor, if cluster c_i and topic z_i agree, the intuition is that the terms in c_i (union of all frequent 2-patterns of c_i) should appear as top terms under z_i (i.e., ranked top according to the term probability under z_i). We define the agreement using symmetrised KL-Divergence between the two distributions ($DIST_c$ and $DIST_z$) corresponding to c_i and z_i respectively. As there is no prior preference on the terms of c_i, we use the uniform distribution over all terms in c_i for $DIST_c$. For $DIST_z$, as only top 20 terms under z_i are usually reliable, we use these top terms

with their probabilities (re-normalized) to represent the topic. Note that a smoothing probability (i.e., a very small value) is also given to every term for calculating KL-Divergence. Given $DIST_c$ and $DIST_z$, the agreement is computed with:

$$Agreement(c, z) = \frac{1}{KL(DIST_c, DIST_z)} \quad (2)$$

The rationale of Equation 2 is that the lesser divergence between $DIST_c$ and $DIST_z$ implies the more agreement between c_i and z_i.

We further employ the *Generalized Plya urn* (GPU) model (Mahmoud, 2008) which was shown to be effective in leveraging semantically related words (Chen et al., 2013a, Chen et al., 2013b, Mimno et al., 2011). The GPU model here basically states that assigning topic z_i and cluster c_i to term w_i will not only increase the probability of connecting z_i and c_i with w_i, but also make it more likely to associate z_i and c_i with term w' where w' shares a 2-pattern with w_i in c_i. The amount of probability increase is determined by matrix $\mathbb{A}_{c,w',w}$ defined as:

$$\mathbb{A}_{c,w',w} = \begin{cases} 1, & \text{if } w = w' \\ \sigma, & \text{if } (w, w') \in c, w \neq w' \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

where value 1 controls the probability increase of w by seeing w itself, and σ controls the probability increase of w' by seeing w. Please refer to (Chen et al., 2013b) for more details.

Putting together Equations 1, 2 and 3 into a blocked Gibbs Sampler, we can define the following sampling distribution in Gibbs sampler so that it provides helpful guidance in determining the usefulness of the prior knowledge and in selecting the semantically coherent topic.

$$\begin{aligned} &P(z_i = t, c_i = c | \boldsymbol{z}^{-i}, \boldsymbol{c}^{-i}, \boldsymbol{w}, \alpha, \beta, \gamma, \mathbb{A}) \\ &\propto \sum_{(w,w') \in c} Co\text{-}Doc(w, w') \times Agreement(c, t) \\ &\times \frac{n^{-i}_{m,t} + \alpha}{\sum_{t'=1}^{T}(n^{-i}_{m,t'} + \alpha)} \\ &\times \frac{\sum_{w'=1}^{V}\sum_{v'=1}^{V} \mathbb{A}_{c,v',w'} \times n^{-i}_{t,c,v'} + \beta}{\sum_{c'=1}^{C}(\sum_{w'=1}^{V}\sum_{v'=1}^{V} \mathbb{A}_{c',v',w'} \times n^{-i}_{t,c',v'} + \beta)} \\ &\times \frac{\sum_{w'=1}^{V} \mathbb{A}_{c,w',w_i} \times n^{-i}_{t,c,w'} + \gamma}{\sum_{v'=1}^{V}(\sum_{w'=1}^{V} \mathbb{A}_{c,w',v'} \times n^{-i}_{t,c,w'} + \gamma)} \end{aligned} \quad (4)$$

where n^{-i} denotes the count excluding the current assignment of z_i and c_i, i.e., $\boldsymbol{z}^{-i}$ and $\boldsymbol{c}^{-i}$. $n_{m,t}$ denotes the number of times that topic t was assigned to terms in document m. $n_{t,c}$ denotes the times that cluster c occurs under topic t. $n_{t,c,v}$ refers to the number of times that term v appears in cluster c under topic t. α, β and γ are predefined Dirichlet hyperparameters.

Note that although the above Gibbs sampler is able to distinguish useful knowledge from wrong knowledge, it is possible that there is no cluster corroborates for a particular term. For every term w, apart from its knowledge clusters, we also add a singleton cluster for w, i.e., a cluster with one pattern $\{w, w\}$ only. When no knowledge cluster is applicable, this singleton cluster is used. As a singleton cluster does not contain any knowledge information but only the word itself, Equations 1 and 2 cannot be computed. For the values of singleton clusters for these two equations, we assign them as the averages of those values of all non-singleton knowledge clusters.

6 Experiments

This section evaluates and compares the proposed AKL model with three baseline models LDA, MC-LDA, and GK-LDA. LDA (Blei et al., 2003) is the most popular unsupervised topic model. MC-LDA (Chen et al., 2013b) is a recent knowledge-based model for aspect extraction. GK-LDA (Chen et al., 2013a) handles wrong knowledge by setting prior weights using the ratio of word probabilities. Our automatically extracted knowledge is provided to these models. Note that cannot-set of MC-LDA is not used in AKL.

6.1 Experimental Settings

Dataset. We created a large dataset containing reviews from 36 product domains or types from Amazon.com. The product domain names are listed in Table 1. Each domain contains $1,000$ reviews. This gives us 36 domain corpora. We have made the dataset publically available at the website of the first author.

Pre-processing. We followed (Chen et al., 2013b) to employ standard pre-processing like lemmatization and stopword removal. To have a fair comparison, we also treat each sentence as a document as in (Chen et al., 2013a, Chen et al., 2013b).

Parameter Settings. For all models, posterior estimates of latent variables were taken with a sampling lag of 20 iterations in the post burn-in phase (first 200 iterations for burn-in) with $2,000$ iterations in total. The model parameters were tuned on the development set in our pilot experiments

Amplifier	DVD Player	Kindle	MP3 Player	Scanner	Video Player
Blu-Ray Player	GPS	Laptop	Network Adapter	Speaker	Video Recorder
Camera	Hard Drive	Media Player	Printer	Subwoofer	Watch
CD Player	Headphone	Microphone	Projector	Tablet	Webcam
Cell Phone	Home Theater System	Monitor	Radar Detector	Telephone	Wireless Router
Computer	Keyboard	Mouse	Remote Control	TV	Xbox

Table 1: List of 36 domain names.

and set to $\alpha = 1$, $\beta = 0.1$, $T = 15$, and $\sigma = 0.2$. Furthermore, for each cluster, γ is set proportional to the number of terms in it. The other parameters for MC-LDA and GK-LDA were set as in their original papers. For parameters of AKL, we used the top 15 terms for each topic in the clustering phrase. The number of clusters is set to the number of domains. We will test the sensitivity of these clustering parameters in Section 6.4. The minimum support count for frequent pattern mining was set empirically to $min(5, 0.4 \times \#\mathbb{T})$, where $\#\mathbb{T}$ is the number of transactions (i.e., the number of topics from all domains) in a cluster.

Test Settings: We use two test settings as below:

1. (Sections 6.2, 6.3 and 6.4) Test on the same corpora as those used in learning the prior knowledge. This is meaningful as the learning phrase is automatic and unsupervised (Figure 1). We call this *self-learning-and-improvement*.
2. (Section 6.5) Test on new/unseen domain corpora after knowledge learning.

6.2 Topic Coherence

This sub-section evaluates the topics/aspects generated by each model based on Topic Coherence (Mimno et al., 2011) in test setting 1. Traditionally, topic models have been evaluated using perplexity. However, perplexity on the held-out test set does not reflect the semantic coherence of topics and may be contrary to human judgments (Chang et al., 2009). Instead, the metric Topic Coherence has been shown in (Mimno et al., 2011) to correlate well with human judgments. Recently, it has become a standard practice to use Topic Coherence for evaluation of topic models (Arora et al., 2013). A higher Topic Coherence value indicates a better topic interpretability, i.e., semantically more coherent topics.

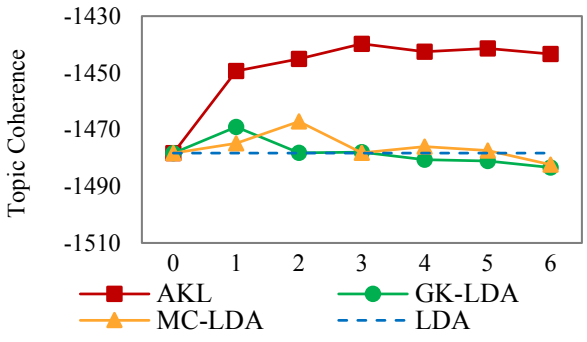


Figure 3: Average Topic Coherence of each model at different learning iterations (Iteration 0 is equivalent to LDA).

Figure 3 shows the average Topic Coherence of each model using knowledge learned at different learning iterations (Figure 1). For MC-LDA or GK-LDA, this is done by replacing AKL in lines 7 and 19 of Figure 1 with MC-LDA or GK-LDA. Each value is the average over all 36 domains. From Figure 3, we can observe the followings:

1. AKL performs the best with the highest Topic Coherence values at all iterations. It is actually the best in all 36 domains. These show that AKL finds more interpretable topics than the baselines. Its values stabilize after iteration 3.
2. Both GK-LDA and MC-LDA perform slightly better than LDA in iterations 1 and 2. MC-LDA does not handle wrong knowledge. This shows that the mined knowledge is of good quality. Although GK-LDA uses large word probability differences under a topic to detect wrong lexical knowledge, it is not as effective as AKL. The reason is that as the lexical knowledge is from general dictionaries rather than mined from relevant domain data, the words in a wrong piece of knowledge usually have a very large probability difference under a topic. However, our knowledge is mined from top words in related topics including topics from the current domain. The words in a piece of incorrect (or correct) knowledge often have similar probabilities under a topic. The proposed dynamic knowledge adjusting mechanism in AKL is superior.

Paired t-test shows that AKL outperforms all baselines significantly ($p < 0.0001$).

6.3 User Evaluation

As our objective is to discover more coherent aspects, we recruited two human judges. Here we also use the test setting 1. Each topic is annotated as coherent if the judge feels that most of its top

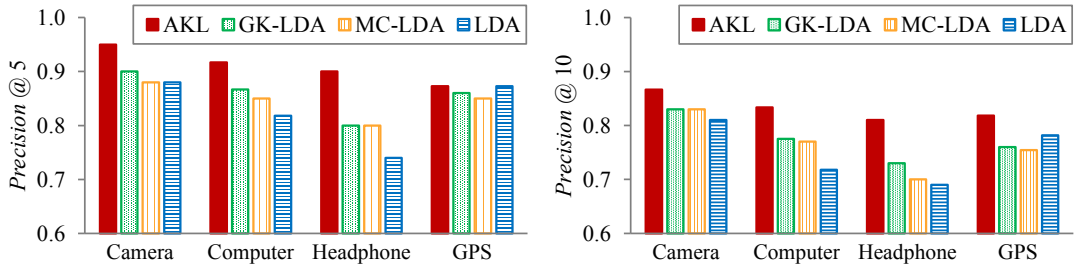


Figure 4: Average $Precision@5$ (Left) and $Precision@10$ (Right) of coherent topics from four models in each domain. (*Headphone* has a lot of overlapping topics in other domains while *GPS* has little.)

terms coherently represent a real-world product aspect; otherwise incoherent. For a coherent topic, each top term is annotated as correct if it reflects the aspect represented by the topic; otherwise incorrect. We labeled the topics of each model at learning iteration 1 where the same pieces of knowledge (extracted from LDA topics at learning iteration 0) are provided to each model. After learning iteration 1, the gap between AKL and the baseline models tends to widen. To be consistent, the results later in Sections 6.4 and 6.5 also show each model at learning iteration 1. We also notice that after a few learning iterations, the topics from AKL model tend to have some resemblance across domains. We found that AKL with 2 learning iterations achieved the best topics. Note that LDA cannot use any prior knowledge.

We manually labeled results from four domains, i.e., Camera, Computer, Headphone, and GPS. We chose Headphone as it has a lot of overlapping of topics with other domains because many electronic products use headphone. GPS was chosen because it does not have much topic overlapping with other domains as its aspects are mostly about Navigation and Maps. Domains Camera and Computer lay in between. We want to see how domain overlapping influences the performance of AKL. Cohen's Kappa scores for annotator agreement are 0.918 (for topics) and 0.872 (for terms).

To measure the results, we compute $Precision@n$ (or $p@n$) based on the annotations, which was also used in (Chen et al., 2013b, Mukherjee and Liu, 2012).

Figure 4 shows the $precision@n$ results for $n = 5$ and 10. We can see that AKL makes improvements in all 4 domains. The improvement varies in domains with the most increase in Headphone and the least in GPS as Headphone overlaps more with other domains than GPS. Note that if a domain shares aspects with many other domains, its model should benefit more; otherwise, it is reasonable to expect lesser improvements. For the baselines, GK-LDA and MC-LDA perform similarly to LDA with minor variations, all of which are inferior to AKL. AKL's improvements over other models are statistically significant based on paired t-test ($p < 0.002$).

In terms of the number of *coherent* topics, AKL discovers one more coherent topic than LDA in Computer and one more coherent topic than GK-LDA and MC-LDA in Headphone. For the other domains, the numbers of coherent topics are the same for all models.

Table 2 shows an example aspect (*battery*) and its top 10 terms produced by AKL and LDA for each domain to give a flavor of the kind of improvements made by AKL. The results for GK-LDA and MC-LDA are about the same as LDA (see also Figure 4). Table 2 focuses on the aspects generated by AKL and LDA. From Table 2, we can see that AKL discovers more correct and meaningful aspect terms at the top. Note that those terms marked in red and italicized are errors. Apart from Table 2, many aspects are dramatically improved by AKL, including some commonly shared aspects such as *Price*, *Screen*, and *Customer Service*.

6.4 Sensitivity to Clustering Parameters

This sub-section investigates the sensitivity of the clustering parameters of AKL (again in test setting 1). The top sub-figure in Figure 5 shows the average Topic Coherence values versus the top k terms per topic used in topic clustering (Section 4.1). The number of clusters is set to the number of domains (see below). We can observe that using $k = 15$ top terms gives the highest value. This is intuitive as too few (or too many) top terms may generate insufficient (or noisy) knowledge.

The bottom sub-figure in Figure 5 shows the average Topic Coherence given different number

Camera		Computer		Headphone		GPS	
AKL	**LDA**	**AKL**	**LDA**	**AKL**	**LDA**	**AKL**	**LDA**
battery	battery	battery	battery	hour	long	battery	*trip*
life	*card*	hour	*cable*	long	battery	hour	battery
hour	*memory*	life	*speaker*	battery	hour	long	hour
long	life	long	*dvi*	life	*comfortable*	*model*	*mile*
charge	usb	*speaker*	*sound*	charge	*easy*	life	long
extra	hour	*sound*	hour	amp	*uncomfortable*	charge	life
minute	minute	charge	*connection*	*uncomfortable*	*headset*	*trip*	*destination*
charger	*sd*	*dvi*	life	*comfortable*	life	*purchase*	*phone*
short	extra	*tv*	*hdmus*	period	*money*	*older*	charge
aa	*device*	*hdmus*	*tv*	*output*	*hard*	*compass*	*mode*

Table 2: Example aspect *Battery* from AKL and LDA in each domain. Errors are italicized in red.

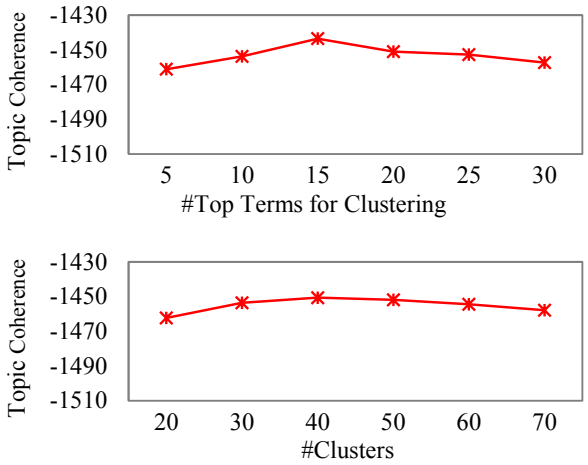


Figure 5: Average topic coherence of AKL versus #top k terms (Top) and #clusters (Bottom).

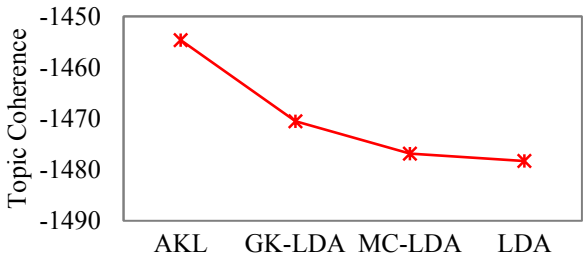


Figure 6: Average topic coherence of each model tested on new/unseen domain.

of clusters. We fix the number of top terms per topic to 15 as it yields the best result (see the top sub-figure in Figure 5). We can see that the performance is not very sensitive to the number of clusters. The model performs similarly for 30 to 50 clusters, with lower Topic Coherence for less than 30 or more than 50 clusters. The significance test indicates that using 30, 40, and 50 clusters, AKL achieved significant improvements over all baseline models ($p < 0.0001$). With more domains, we should expect a larger number of clusters. However, it is difficult to obtain the optimal number of clusters. Thus, we empirically set the number of clusters to the number of domains in our experiments. Note that the number of clusters (C) is expected to be larger than the number of topics in one domain (T) because C is for all domains while T is for one particular domain.

6.5 Test on New Domains

We now evaluate AKL in test setting 2, i.e., the automatically extracted knowledge $\boldsymbol{K}$ (Figure 1) is applied in new/unseen domains other than those in domains $\boldsymbol{D_L}$ used in knowledge learning. The aim is to see how $\boldsymbol{K}$ can help modeling in an unseen domain. In this set of experiments, each domain is tested by using the learned knowledge from the rest 35 domains. Figure 6 shows the average Topic Coherence of each model. The values are also averaged over the 36 tested domains. We can see that AKL achieves the highest Topic Coherence value while LDA has the lowest. The improvements of AKL over all baseline models are significant with $p < 0.0001$.

7 Conclusions

This paper proposed an advanced aspect extraction framework which can learn knowledge automatically from a large number of review corpora and exploit the learned knowledge in extracting more coherent aspects. It first proposed a technique to learn knowledge automatically by clustering and FPM. Then a new topic model with an advanced inference mechanism was proposed to exploit the learned knowledge in a fault-tolerant manner. Experimental results using review corpora from 36 domains showed that the proposed method outperforms state-of-the-art methods significantly.

Acknowledgments

This work was supported in part by a grant from National Science Foundation (NSF) under grant no. IIS-1111092.

References

David Andrzejewski, Xiaojin Zhu, and Mark Craven. 2009. Incorporating domain knowledge into topic modeling via Dirichlet Forest priors. In *Proceedings of ICML*, pages 25–32.

David Andrzejewski, Xiaojin Zhu, Mark Craven, and Benjamin Recht. 2011. A framework for incorporating general domain knowledge into latent Dirichlet allocation using first-order logic. In *Proceedings of IJCAI*, pages 1171–1177.

Sanjeev Arora, Rong Ge, Yonatan Halpern, David Mimno, Ankur Moitra, David Sontag, Yichen Wu, and Michael Zhu. 2013. A Practical Algorithm for Topic Modeling with Provable Guarantees. In *Proceedings of ICML*, pages 280–288.

David M. Blei and Jon D McAuliffe. 2007. Supervised Topic Models. In *Proceedings of NIPS*, pages 121–128.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

S R K Branavan, Harr Chen, Jacob Eisenstein, and Regina Barzilay. 2008. Learning Document-Level Semantic Properties from Free-Text Annotations. In *Proceedings of ACL*, pages 263–271.

Samuel Brody and Noemie Elhadad. 2010. An unsupervised aspect-sentiment model for online reviews. In *Proceedings of NAACL*, pages 804–812.

Giuseppe Carenini, Raymond T Ng, and Ed Zwart. 2005. Extracting knowledge from evaluative text. In *Proceedings of K-CAP*, pages 11–18.

Jonathan Chang, Jordan Boyd-Graber, Wang Chong, Sean Gerrish, and David Blei, M. 2009. Reading Tea Leaves: How Humans Interpret Topic Models. In *Proceedings of NIPS*, pages 288–296.

Zhiyuan Chen, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013a. Discovering Coherent Topics Using General Knowledge. In *Proceedings of CIKM*, pages 209–218.

Zhiyuan Chen, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013b. Exploiting Domain Knowledge in Aspect Extraction. In *Proceedings of EMNLP*, pages 1655–1667.

Zhiyuan Chen, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013c. Leveraging Multi-Domain Prior Knowledge in Topic Models. In *Proceedings of IJCAI*, pages 2071–2077.

Yejin Choi and Claire Cardie. 2010. Hierarchical Sequential Learning for Extracting Opinions and their Attributes. In *Proceedings of ACL*, pages 269–274.

Jacob Eisenstein, Brendan O'Connor, Noah A Smith, and Eric P Xing. 2010. A Latent Variable Model for Geographic Lexical Variation. In *Proceedings of EMNLP*, pages 1277–1287.

Lei Fang and Minlie Huang. 2012. Fine Granular Aspect Analysis using Latent Structural Models. In *Proceedings of ACL*, pages 333–337.

Honglei Guo, Huijia Zhu, Zhili Guo, Xiaoxun Zhang, and Zhong Su. 2009. Product feature categorization with multilevel latent semantic association. In *Proceedings of CIKM*, pages 1087–1096.

Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. 2007. Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery*, 15(1):55–86.

Yulan He, Chenghua Lin, and Harith Alani. 2011. Automatically Extracting Polarity-Bearing Topics for Cross-Domain Sentiment Classification. In *Proceedings of ACL*, pages 123–131.

Thomas Hofmann. 1999. Probabilistic Latent Semantic Analysis. In *Proceedings of UAI*, pages 289–296.

Minqing Hu and Bing Liu. 2004. Mining and Summarizing Customer Reviews. In *Proceedings of KDD*, pages 168–177.

Yuening Hu, Jordan Boyd-Graber, and Brianna Satinoff. 2011. Interactive Topic Modeling. In *Proceedings of ACL*, pages 248–257.

Jagadeesh Jagarlamudi, Hal Daumé III, and Raghavendra Udupa. 2012. Incorporating Lexical Priors into Topic Models. In *Proceedings of EACL*, pages 204–213.

Niklas Jakob and Iryna Gurevych. 2010. Extracting Opinion Targets in a Single- and Cross-Domain Setting with Conditional Random Fields. In *Proceedings of EMNLP*, pages 1035–1045.

Yohan Jo and Alice H. Oh. 2011. Aspect and sentiment unification model for online review analysis. In *Proceedings of WSDM*, pages 815–824.

Jeon-hyung Kang, Jun Ma, and Yan Liu. 2012. Transfer Topic Modeling with Ease and Scalability. In *Proceedings of SDM*, pages 564–575.

L Kaufman and P J Rousseeuw. 1990. *Finding groups in data: an introduction to cluster analysis*. John Wiley and Sons.

Suin Kim, Jianwen Zhang, Zheng Chen, Alice Oh, and Shixia Liu. 2013. A Hierarchical Aspect-Sentiment Model for Online Reviews. In *Proceedings of AAAI*, pages 526–533.

Nozomi Kobayashi, Kentaro Inui, and Yuji Matsumoto. 2007. Extracting Aspect-Evaluation and Aspect-of Relations in Opinion Mining. In *Proceedings of EMNLP*, pages 1065–1074.

Lun-Wei Ku, Yu-Ting Liang, and Hsin-Hsi Chen. 2006. Opinion Extraction, Summarization and Tracking in News and Blog Corpora. In *Proceedings of AAAI-CAAW*, pages 100–107.

Angeliki Lazaridou, Ivan Titov, and Caroline Sporleder. 2013. A Bayesian Model for Joint Unsupervised Induction of Sentiment, Aspect and Discourse Representations. In *Proceedings of ACL*, pages 1630–1639.

Fangtao Li, Chao Han, Minlie Huang, Xiaoyan Zhu, Yingju Xia, Shu Zhang, and Hao Yu. 2010. Structure-Aware Review Mining and Summarization. In *Proceedings of COLING*, pages 653–661.

Peng Li, Yinglin Wang, Wei Gao, and Jing Jiang. 2011. Generating Aspect-oriented Multi-Document Summarization with Event-aspect model. In *Proceedings of EMNLP*, pages 1137–1146.

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proceedings of CIKM*, pages 375–384.

Kang Liu, Liheng Xu, and Jun Zhao. 2013. Syntactic Patterns versus Word Alignment: Extracting Opinion Targets from Online Reviews. In *Proceedings of ACL*, pages 1754–1763.

Bing Liu. 2012. *Sentiment Analysis and Opinion Mining*. Morgan & Claypool Publishers.

Yue Lu and Chengxiang Zhai. 2008. Opinion integration through semi-supervised topic modeling. In *Proceedings of WWW*, pages 121–130.

Yue Lu, ChengXiang Zhai, and Neel Sundaresan. 2009. Rated aspect summarization of short comments. In *Proceedings of WWW*, pages 131–140.

Bin Lu, Myle Ott, Claire Cardie, and Benjamin K Tsou. 2011. Multi-aspect Sentiment Analysis with Topic Models. In *Proceedings of ICDM Workshops*, pages 81–88.

Yue Lu, Hongning Wang, ChengXiang Zhai, and Dan Roth. 2012. Unsupervised discovery of opposing opinion networks from forum discussions. In *Proceedings of CIKM*, pages 1642–1646.

Hosam Mahmoud. 2008. *Polya Urn Models*. Chapman & Hall/CRC Texts in Statistical Science.

Qiaozhu Mei, Xu Ling, Matthew Wondra, Hang Su, and ChengXiang Zhai. 2007. Topic sentiment mixture: modeling facets and opinions in weblogs. In *Proceedings of WWW*, pages 171–180.

David Mimno, Hanna M. Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of EMNLP*, pages 262–272.

Samaneh Moghaddam and Martin Ester. 2013. The FLDA Model for Aspect-based Opinion Mining: Addressing the Cold Start Problem. In *Proceedings of WWW*, pages 909–918.

Arjun Mukherjee and Bing Liu. 2012. Aspect Extraction through Semi-Supervised Modeling. In *Proceedings of ACL*, pages 339–348.

Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. *IEEE Trans. Knowl. Data Eng.*, 22(10):1345–1359.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

James Petterson, Alex Smola, Tibério Caetano, Wray Buntine, and Shravan Narayanamurthy. 2010. Word Features for Latent Dirichlet Allocation. In *Proceedings of NIPS*, pages 1921–1929.

AM Popescu and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of HLT*, pages 339–346.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2011. Opinion Word Expansion and Target Extraction through Double Propagation. *Computational Linguistics*, 37(1):9–27.

Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D. Manning. 2009. Labeled LDA: a supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of EMNLP*, pages 248–256.

Michal Rosen-Zvi, Chaitanya Chemudugunta, Thomas Griffiths, Padhraic Smyth, and Mark Steyvers. 2010. Learning author-topic models from text corpora. *ACM Transactions on Information Systems*, 28(1):1–38.

Christina Sauper and Regina Barzilay. 2013. Automatic Aggregation by Joint Modeling of Aspects and Values. *J. Artif. Intell. Res. (JAIR)*, 46:89–127.

Swapna Somasundaran and J Wiebe. 2009. Recognizing stances in online debates. In *Proceedings of ACL*, pages 226–234.

Ivan Titov and Ryan McDonald. 2008. A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of ACL*, pages 308–316.

Hongning Wang, Yue Lu, and Chengxiang Zhai. 2010. Latent aspect rating analysis on review text data: a rating regression approach. In *Proceedings of KDD*, pages 783–792.

Yuanbin Wu, Qi Zhang, Xuanjing Huang, and Lide Wu. 2009. Phrase dependency parsing for opinion mining. In *Proceedings of EMNLP*, pages 1533–1541.

Liheng Xu, Kang Liu, Siwei Lai, Yubo Chen, and Jun Zhao. 2013. Mining Opinion Words and Opinion Targets in a Two-Stage Framework. In *Proceedings of ACL*, pages 1764–1773.

GR Xue, Wenyuan Dai, Q Yang, and Y Yu. 2008. Topic-bridged PLSA for cross-domain text classification. In *Proceedings of SIGIR*, pages 627–634.

Bishan Yang and Claire Cardie. 2013. Joint Inference for Fine-grained Opinion Extraction. In *Proceedings of ACL*, pages 1640–1649.

Shuang Hong Yang, Steven P. Crain, and Hongyuan Zha. 2011. Bridging the language gap: Topic adaptation for documents with different technicality. In *Proceedings of AISTATS*, pages 823–831.

Jianxing Yu, Zheng-Jun Zha, Meng Wang, and Tat-Seng Chua. 2011. Aspect Ranking: Identifying Important Product Aspects from Online Consumer Reviews. In *Proceedings of ACL*, pages 1496–1505.

Zhongwu Zhai, Bing Liu, Hua Xu, and Peifa Jia. 2011. Constrained LDA for grouping product features in opinion mining. In *Proceedings of PAKDD*, pages 448–459.

Wayne Xin Zhao, Jing Jiang, Hongfei Yan, and Xiaoming Li. 2010. Jointly Modeling Aspects and Opinions with a MaxEnt-LDA Hybrid. In *Proceedings of EMNLP*, pages 56–65.

Yanyan Zhao, Bing Qin, and Ting Liu. 2012. Collocation polarity disambiguation using web-based pseudo contexts. In *Proceedings of EMNLP-CoNLL*, pages 160–170.

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2013. Collective Opinion Target Extraction in Chinese Microblogs. In *Proceedings of EMNLP*, pages 1840–1850.

Li Zhuang, Feng Jing, and Xiao-Yan Zhu. 2006. Movie review mining and summarization. In *Proceedings of CIKM*, pages 43–50.

Anchors Regularized: Adding Robustness and Extensibility to Scalable Topic-Modeling Algorithms

Thang Nguyen
iSchool and UMIACS,
University of Maryland
and National Library of Medicine,
National Institutes of Health
daithang@umiacs.umd.edu

Yuening Hu
Computer Science
University of Maryland
ynhu@cs.umd.edu

Jordan Boyd-Graber
iSchool and UMIACS
University of Maryland
jbg@umiacs.umd.edu

Abstract

Spectral methods offer scalable alternatives to Markov chain Monte Carlo and expectation maximization. However, these new methods lack the rich priors associated with probabilistic models. We examine Arora et al.'s anchor words algorithm for topic modeling and develop new, regularized algorithms that not only mathematically resemble Gaussian and Dirichlet priors but also improve the interpretability of topic models. Our new regularization approaches make these efficient algorithms more flexible; we also show that these methods can be combined with informed priors.

1 Introduction

Topic models are of practical and theoretical interest. Practically, they have been used to understand political perspective (Paul and Girju, 2010), improve machine translation (Eidelman et al., 2012), reveal literary trends (Jockers, 2013), and understand scientific discourse (Hall et al., 2008). Theoretically, their latent variable formulation has served as a foundation for more robust models of other linguistic phenomena (Brody and Lapata, 2009).

Modern topic models are formulated as a latent variable model. Like hidden Markov models (Rabiner, 1989, HMM), each token comes from one of K unknown distributions. Unlike a HMM, topic models assume that each document is an *admixture* of these hidden components called topics. Posterior inference discovers the hidden variables that best explain a dataset. Typical solutions use MCMC (Griffiths and Steyvers, 2004) or variational EM (Blei et al., 2003), which can be viewed as local optimization: searching for the latent variables that maximize the data likelihood.

An exciting vein of new research provides provable polynomial-time alternatives. These approaches provide solutions to hidden Markov models (Anandkumar et al., 2012), mixture models (Kannan et al., 2005), and latent variable grammars (Cohen et al., 2013). The key insight is not to directly optimize observation likelihood but to instead discover latent variables that can reconstruct statistics of the assumed generative model. Unlike search-based methods, which can be caught in local minima, these techniques are often guaranteed to find global optima.

These general techniques can be improved by making reasonable assumptions about the models. For example, Arora et al. (2012b)'s approach for inference in topic models assume that each topic has a unique "anchor" word (thus, we call this approach **anchor**). This approach is fast and effective; because it only uses word co-occurrence information, it can scale to much larger datasets than MCMC or EM alternatives. We review the **anchor** method in Section 2.

Despite their advantages, these techniques are not a panacea. They do not accommodate the rich priors that modelers have come to expect. Priors can improve performance (Wallach et al., 2009), provide domain adaptation (Daumé III, 2007; Finkel and Manning, 2009), and guide models to reflect users' needs (Hu et al., 2013). In Section 3, we regularize the **anchor** method to trade-off the reconstruction fidelity with the penalty terms that mimic Gaussian and Dirichlet priors.

Another shortcoming is that these models have not been scrutinized using standard NLP evaluations. Because these approaches emerged from the theory community, **anchor**'s evaluations, when present, typically use training reconstruction. In Section 4, we show that our regularized models can generalize to previously unseen data—as measured by held-out likelihood (Blei et al., 2003)—and are more interpretable (Chang et al., 2009; Newman et al., 2010). We also show that our extension to the **anchor** method enables new applications: for

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 359–369,
Baltimore, Maryland, USA, June 23-25 2014.

K	number of topics
V	vocabulary size
M	document frequency: minimum documents an anchor word candidate must appear in
$\boldsymbol{Q}$	word co-occurrence matrix $Q_{i,j} = p(w_1 = i, w_2 = j)$
$\bar{\boldsymbol{Q}}$	conditional distribution of Q $\bar{Q}_{i,j} = p(w_1 = j \mid w_2 = i)$
$\bar{\boldsymbol{Q}}_{i,\cdot}$	row i of $\bar{Q}$
$\boldsymbol{A}$	topic matrix, of size $V \times K$ $A_{j,k} = p(w = j \mid z = k)$
$\boldsymbol{C}$	anchor coefficient of size $K \times V$ $C_{j,k} = p(z = k \mid w = j)$
$\mathcal{S}$	set of anchor word indexes $\{s_1, \ldots s_K\}$
λ	regularization weight

Table 1: Notation used. Matrices are in bold ($\boldsymbol{Q}, \boldsymbol{C}$), sets are in script $\mathcal{S}$

example, using an informed priors to discover concepts of interest.

Having shown that regularization does improve performance, in Section 5 we explore why. We discuss the trade-off of training data reconstruction with sparsity and why regularized topics are more interpretable.

2 Anchor Words: Scalable Topic Models

In this section, we briefly review the **anchor** method and place it in the context of topic model inference. Once we have established the **anchor** objective function, in the next section we regularize the objective function.

Rethinking Data: Word Co-occurrence Inference in topic models can be viewed as a black box: given a set of documents, discover the topics that best explain the data. The difference between **anchor** and conventional inference is that while conventional methods take a collection of documents as input, **anchor** takes *word co-occurrence* statistics. Given a vocabulary of size V, we represent this joint distribution as $\boldsymbol{Q}_{i,j} = p(w_1 = i, w_2 = j)$, each cell represents the probability of words appearing together in a document.

Like other topic modeling algorithms, the output of the **anchor** method is the topic word distributions $\boldsymbol{A}$ with size $V * K$, where K is the total number of topics desired, a parameter of the algorithm. The k^{th} column of A will be the topic distribution over all words for topic k, and $\boldsymbol{A}_{w,k}$ is the probability of observing type w given topic k.

Anchors: Topic Representatives The **anchor** method (Arora et al., 2012a) is based on the separability assumption (Donoho and Stodden, 2003), which assumes that each topic contains at least one namesake "anchor word" that has non-zero probability only in that topic. Intuitively, this means that each topic has unique, specific word that, when used, identifies that topic. For example, while "run", "base", "fly", and "shortstop" are associated with a topic about baseball, only "shortstop" is unambiguous, so it could serve as this topic's anchor word.

Let's assume that we knew what the anchor words were: a set $\mathcal{S}$ that indexes rows in $\boldsymbol{Q}$. Now consider the **conditional distribution** of word i, the probability of the rest of the vocabulary given an observation of word i; we represent this as $\bar{\boldsymbol{Q}}_{i,\cdot}$, as we can construct this by normalizing the rows of $\boldsymbol{Q}$. For an anchor word $s_a \in \mathcal{S}$, this will look like a topic; $\bar{\boldsymbol{Q}}_{\text{"shortstop"},\cdot}$ will have high probability for words associated with baseball.

The key insight of the **anchor** algorithm is that the conditional distribution of polysemous non-anchor words can be reconstructed as a linear combination of the conditional distributions of anchor words. For example, $\bar{\boldsymbol{Q}}_{\text{"fly"},\cdot}$ could be reconstructed by combining the anchor words "insecta", "boeing", and "shortshop". We represent the coefficients of this reconstruction as a matrix $\boldsymbol{C}$, where $C_{i,k} = p(z = k \mid w = i)$. Thus, for any word i,

$$\bar{\boldsymbol{Q}}_{i,\cdot} \approx \sum_{s_k \in \mathcal{S}} C_{i,k} \bar{\boldsymbol{Q}}_{s_k,\cdot} \tag{1}$$

The coefficient matrix is **not** the usual output of a topic modeling algorithm. The usual output is the probability of a word *given a topic*. The coefficient matrix C is the probability of a topic *given a word*. We use Bayes rule to recover the topic distribution $p(w = i|z = k) \equiv$

$$\begin{aligned} A_{i,k} &\propto p(z = k|w = i)p(w = i) \\ &= C_{i,k} \sum_j \bar{Q}_{i,j} \end{aligned} \tag{2}$$

where $p(w)$ is the normalizer of $\boldsymbol{Q}$ to obtain $\bar{\boldsymbol{Q}}_{w,\cdot}$.

The geometric argument for finding the anchor words is one of the key contributions of Arora et al. (2012a) and is beyond the scope of this paper. The algorithms in Section 3 use the anchor selection subroutine unchanged. The difference in our approach is in how we discover the anchor coefficients C.

From Anchors to Topics After we have the anchor words, we need to find the coefficients that

best reconstruct the data $\bar{Q}$ (Equation 1). Arora et al. (2012a) chose the C that minimizes the KL divergence between $\bar{Q}_{i,\cdot}$ and the reconstruction based on the anchor word's conditional word vectors $\sum_{s_k \in \mathcal{S}} C_{i,k} \bar{\boldsymbol{Q}}_{s_k,\cdot}$,

$$\boldsymbol{C}_{i,\cdot} = \text{argmin}_{\boldsymbol{C}_{i,\cdot}} D_{\text{KL}} \left(\bar{\boldsymbol{Q}}_{i,\cdot} \,||\, \sum_{s_k \in S} C_{i,k} \bar{\boldsymbol{Q}}_{s_k,\cdot} \right). \tag{3}$$

The **anchor** method is fast, as it only depends on the size of the vocabulary once the co-occurrence statistics $\boldsymbol{Q}$ are obtained. However, it does not support rich priors for topic models, while MCMC (Griffiths and Steyvers, 2004) and variational EM (Blei et al., 2003) methods can. This prevents models from using priors to guide the models to discover particular themes (Zhai et al., 2012), or to encourage sparsity in the models (Yao et al., 2009). In the rest of this paper, we correct this lacuna by adding regularization inspired by Bayesian priors to the **anchor** algorithm.

3 Adding Regularization

In this section, we add regularizers to the **anchor** objective (Equation 3). In this section, we briefly review regularizers and then add two regularizers, inspired by Gaussian (L_2, Section 3.1) and Dirichlet priors (Beta, Section 3.2), to the **anchor** objective function (Equation 3).

Regularization terms are ubiquitous. They typically appear as an additional term in an optimization problem. Instead of optimizing a function just of the data x and parameters β, $f(x, \beta)$, one optimizes an objective function that includes a regularizer that is only a function of parameters: $f(w, \beta) + r(\beta)$. Regularizers are critical in staid methods like linear regression (Ng, 2004), in workhorse methods such as maximum entropy modeling (Dudík et al., 2004), and also in emerging fields such as deep learning (Wager et al., 2013).

In addition to being useful, regularization terms are appealing theoretically because they often correspond to probabilistic interpretations of parameters. For example, if we are seeking the MLE of a probabilistic model parameterized by β, $p(x|\beta)$, adding a regularization term $r(\beta) = \sum_{i=1}^{L} \beta_i^2$ corresponds to adding a Gaussian prior

$$f(\beta_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \text{exp} \left\{ -\frac{\beta_i^2}{2\sigma^2} \right\} \tag{4}$$

Corpus	Train	Dev	Test	Vocab
NIPS	1231	247	262	12182
20NEWS	11243	3760	3726	81604
NYT	9255	2012	1959	34940

Table 2: The number of documents in the train, development, and test folds in our three datasets.

and maximizing log probability of the posterior (ignoring constant terms) (Rennie, 2003).

3.1 L_2 Regularization

The simplest form of regularization we can add is L_2 regularization. This is similar to assuming that probability of a word given a topic comes from a Gaussian distribution. While the distribution over topics is typically Dirichlet, Dirichlet distributions have been replaced by logistic normals in topic modeling applications (Blei and Lafferty, 2005) and for probabilistic grammars of language (Cohen and Smith, 2009).

Augmenting the **anchor** objective with an L_2 penalty yields

$$\begin{aligned} \boldsymbol{C}_{i,\cdot} = &\text{argmin}_{C_{i,\cdot}} D_{\text{KL}} \left(\bar{\boldsymbol{Q}}_{i,\cdot} \,||\, \sum_{s_k \in S} C_{i,k} \bar{\boldsymbol{Q}}_{s_k,\cdot} \right) \\ &+ \lambda \|\boldsymbol{C}_{i,\cdot} - \mu_{i,\cdot}\|_2^2, \end{aligned} \tag{5}$$

where regularization weight λ balances the importance of a high-fidelity reconstruction against the regularization, which encourages the anchor coefficients to be close to the vector μ. When the mean vector μ is zero, this encourages the topic coefficients to be zero. In Section 4.3, we use a non-zero mean μ to encode an informed prior to encourage topics to discover specific concepts.

3.2 Beta Regularization

The more common prior for topic models is a Dirichlet prior (Minka, 2000). However, we cannot apply this directly because the optimization is done on a row-by-row basis of the anchor coefficient matrix $\boldsymbol{C}$, optimizing $\boldsymbol{C}$ for a fixed word w for and all topics. If we want to model the probability of a word, it must be the probability of word w in a topic versus all other words.

Modeling this dichotomy (one versus all others in a topic) is possible. The constructive definition of the Dirichlet distribution (Sethuraman, 1994) states that if one has a V-dimensional multinomial $\theta \sim \text{Dir}(\alpha_1 \ldots \alpha_V)$, then the marginal distribution

of θ_w follows $\theta_w \sim \text{Beta}(\alpha_w, \sum_{i \neq w} \alpha_i)$. This is the tool we need to consider the distribution of a single word's probability.

This requires including the topic matrix as part of the objective function. The topic matrix is a linear transformation of the coefficient matrix (Equation 2). The objective for beta regularization becomes

$$\boldsymbol{C}_{i,\cdot} = \text{argmin}_{C_{i,\cdot}} D_{\text{KL}} \left(\bar{\boldsymbol{Q}}_{i,\cdot} \mid\mid \sum_{s_k \in S} C_{i,k} \bar{\boldsymbol{Q}}_{s_k,\cdot} \right) - \lambda \sum_{s_k \in S} \log \left(\text{Beta}(A_{i,k}; a, b) \right), \quad (6)$$

where λ again balances reconstruction against the regularization. To ensure the tractability of this algorithm, we enforce a convex regularization function, which requires that $a > 1$ and $b > 1$. If we enforce a uniform prior—$\mathbb{E}_{\text{Beta}(a,b)}[A_{i,k}] = \frac{1}{V}$—and that the *mode* of the distribution is also $\frac{1}{V}$,[1] this gives us the following parametric form for a and b:

$$a = \frac{x}{V} + 1, \text{ and } b = \frac{(V-1)x}{V} + 1 \quad (7)$$

for real x greater than zero.

3.3 Initialization and Convergence

Equation 5 and Equation 6 are optimized using L-BFGS gradient optimization (Galassi et al., 2003). We initialize $\boldsymbol{C}$ randomly from $\text{Dir}(\alpha)$ with $\alpha = \frac{60}{V}$ (Wallach et al., 2009). We update $\boldsymbol{C}$ after optimizing all V rows. The newly updated $\boldsymbol{C}$ replaces the old topic coefficients. We track how much the topic coefficients $\boldsymbol{C}$ change between two consecutive iterations i and $i+1$ and represent it as $\Delta \boldsymbol{C} \equiv \|\boldsymbol{C}^{i+1} - \boldsymbol{C}^{i}\|_2$. We stop optimization when $\Delta \boldsymbol{C} \leq \delta$. When $\delta = 0.1$, the L_2 and unregularized anchor algorithm converges after a single iteration, while beta regularization typically converges after fewer than ten iterations (Figure 4).

4 Regularization Improves Topic Models

In this section, we measure the performance of our proposed regularized anchor word algorithms. We will refer to specific algorithms in bold. For example, the original anchor algorithm is **anchor**. Our L_2 regularized variant is **anchor-L_2**, and our beta regularized variant is **anchor-beta**. To provide conventional baselines, we also compare our methods against topic models from variational inference (Blei et al., 2003, **variational**) and MCMC (Griffiths and Steyvers, 2004; McCallum, 2002, **MCMC**).

We apply these inference strategies on three diverse corpora: scientific articles from the Neural Information Processing Society (NIPS),[2] Internet newsgroups postings (20NEWS),[3] and New York Times editorials (Sandhaus, 2008, NYT). Statistics for the datasets are summarized in Table 2. We split each dataset into a training fold (70%), development fold (15%), and a test fold (15%): the training data are used to fit models; the development set are used to select parameters (anchor threshold M, document prior α, regularization weight λ); and final results are reported on the test fold.

We use two evaluation measures, held-out likelihood (Blei et al., 2003, **HL**) and topic interpretability (Chang et al., 2009; Newman et al., 2010, **TI**). Held-out likelihood measures how well the model can reconstruct held-out documents that the model has never seen before. This is the typical evaluation for probabilistic models. Topic interpretability is a more recent metric to capture how useful the topics can be to human users attempting to make sense of a large datasets.

Held-out likelihood cannot be computed with existing **anchor** algorithms, so we use the topic distributions learned from **anchor** as input to a reference variational inference implementation (Blei et al., 2003) to compute **HL**. This requires an additional parameter, the Dirichlet prior α for the per-document distribution over topics. We select α using grid search on the development set.

To compute **TI** and evaluate topic coherence, we use normalized pairwise mutual information (NPMI) (Lau et al., 2014) over topics' twenty most probable words. Topic coherence is computed against the NPMI of a reference corpus. For coherence evaluations, we use both intrinsic and extrinsic text collections to compute NPMI. Intrinsic coherence (TI-i) is computed on training and development data at development time and on training and test data at test time. Extrinsic coherence (TI-e) is computed from English Wikipedia articles, with disjoint halves (1.1 million pages each) for distinct development and testing TI-e evaluation.

[1] For $a, b < 1$, the expected value is still the uniform distribution but the mode lies at the boundaries of the simplex. This corresponds to a sparse Dirichlet distribution, which our optimization cannot at present model.

[2] http://cs.nyu.edu/~roweis/data.html

[3] http://qwone.com/~jason/20Newsgroups/

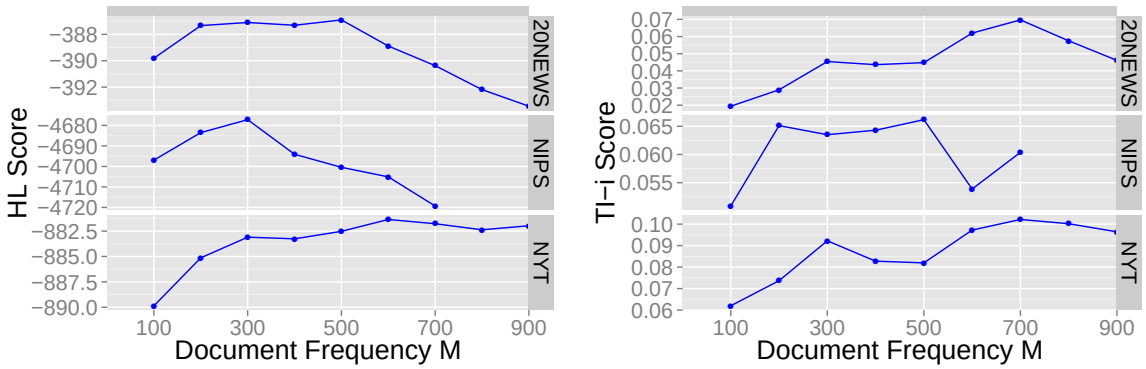


Figure 1: Grid search for document frequency M for our datasets with 20 topics (other configurations not shown) on development data. The performance on both HL and TI score indicate that the unregularized **anchor** algorithm is very sensitive to M. The M selected here is applied to subsequent models.

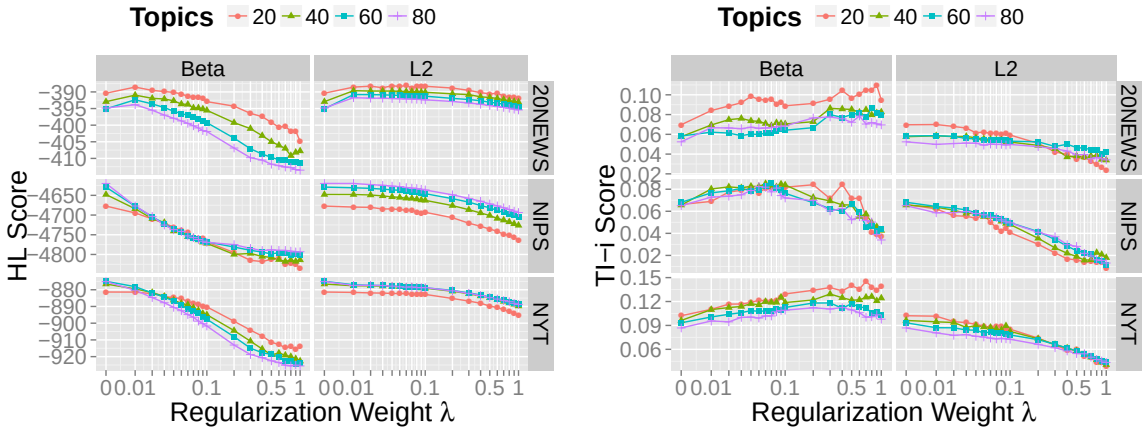


Figure 2: Selection of λ based on HL and TI scores on the development set. The value of $\lambda = 0$ is equivalent to the original **anchor** algorithm; regularized versions find better solutions as the regularization weight λ becomes non-zero.

4.1 Grid Search for Parameters on Development Set

Anchor Threshold A good anchor word must have a unique, specific context but also explain other words well. A word that appears only once will have a very specific cooccurence pattern but will explain other words' coocurrence poorly because the observations are so sparse. As discussed in Section 2, the **anchor** method uses document frequency M as a threshold to only consider words with robust counts.

Because all regularizations benefit equally from higher-quality anchor words, we use cross-validation to select the document frequency cutoff M using the unregularized **anchor** algorithm. Figure 1 shows the performance of **anchor** with different M on our three datasets with 20 topics for our two measures HL and TI-i.

Regularization Weight Once we select a cutoff M for each combination of dataset, number of topics K and a evaluation measure, we select a regularization weight λ on the development set. Figure 2 shows that **beta** regularization framework improves topic interpretability TI-i on all datasets and improved the held-out likelihood HL on 20NEWS. The L_2 regularization also improves held-out likelihood HL for the 20NEWS corpus (Figure 2).

In the interests of space, we do not show the figures for selecting M and λ using TI-e, which is similar to TI-i: **anchor-beta** improves TI-e score on all datasets, **anchor-L_2** improves TI-e on 20NEWS and NIPS with 20 topics and NYT with 40 topics.

4.2 Evaluating Regularization

With document frequency M and regularization weight λ selected from the development set, we

compare the performance of those models on the test set. We also compare with standard implementations of Latent Dirichlet Allocation: Blei's LDAC (**variational**) and Mallet (**mcmc**). We run 100 iterations for LDAC and 5000 iterations for Mallet.

Each result is averaged over three random runs and appears in Figure 3. The highly-tuned, widely-used implementations uniformly have better held-out likelihood than **anchor**-based methods, but the much faster **anchor** methods are often comparable. Within **anchor**-based methods, L_2-regularization offers comparable held-out likelihood as unregularized **anchor**, while **anchor-beta** often has better interpretability. Because of the mismatch between the specialized vocabulary of NIPS and the general-purpose language of Wikipedia, TI-e has a high variance.

4.3 Informed Regularization

A frequent use of priors is to add information to a model. This is not possible with the existing **anchor** method. An informed prior for topic models seeds a topic with words that describe a topic of interest. In a topic model, these seeds will serve as a "magnet", attracting similar words to the topic (Zhai et al., 2012).

We can achieve a similar goal with **anchor-**L_2. Instead of encouraging anchor coefficients to be zero in Equation 5, we can instead encourage word probabilities to close to an arbitrary mean $\mu_{i,k}$. This vector can reflect expert knowledge.

One example of a source of expert knowledge is Linguistic Inquiry and Word Count (Pennebaker and Francis, 1999, LIWC), a dictionary of keywords related to sixty-eight psychological concepts such as positive emotions, negative emotions, and death. For example, it associates "excessive, estate, money, cheap, expensive, living, profit, live, rich, income, poor, etc." for the concept materialism.

We associate each anchor word with its closest LIWC category based on the cooccurrence matrix $\boldsymbol{Q}$. This is computed by greedily finding the anchor word that has the highest cooccurrence score for any LIWC category: we define the score of a category to anchor word w_{s_k} as $\sum_i Q_{s_k,i}$, where i ranges over words in this category; we compute the scores of all categories to all anchor words; then we find the highest score and assign the category to that anchor word; we greedily repeat this process until all anchor words have a category.

Given these associations, we create a goal mean $\mu_{i,k}$. If there are L_i anchor words associated with LIWC word i, $\mu_{i,k} = \frac{1}{L_i}$ if this keyword i is associated with anchor word w_{s_k} and zero otherwise.

We apply **anchor-**L_2 with informed priors on NYT with twenty topics and compared the topics against the original topics from **anchor**. Table 3 shows that the topic with anchor word "soviet", when combined with LIWC, draws in the new words "bush" and "nuclear"; reflecting the threats of force during the cold war. For the topic with topic word "arms", when associated with the LIWC category with the terms "agree" and "agreement", draws in "clinton", who represented a more conciliatory foreign policy compared to his republican predecessors.

5 Discussion

Having shown that regularization can improve the **anchor** topic modeling algorithm, in this section we discuss *why* these regularizations can improve the model and the implications for practitioners.

Efficiency Efficiency is a function of the number of iterations and the cost of each iteration. Both **anchor** and **anchor-**L_2 require a single iteration, although the latter's iteration is slightly more expensive. For **beta**, as described in Section 3.2, we update anchor coefficients C row by row, and then repeat the process over several iterations until it converges. However, it often converges within ten iterations (Figure 4) on all three datasets: this requires much fewer iterations than MCMC or variational inference, and the iterations are less expensive. In addition, since we optimize each row $C_{i,\cdot}$ independently, the algorithm can be easily parallelized.

Sensitivity to Document Frequency While the original **anchor** is sensitive to the document frequency M (Figure 1), adding regularization makes this less critical. Both **anchor-**L_2 and **anchor-beta** are less sensitive to M than **anchor**.

To highlight this, we compare the topics of **anchor** and **anchor-beta** when $M = 100$. As Table 4 shows, the words "article", "write", "don" and "doe" appear in most of **anchor**'s topics. While **anchor-**L_2 also has some bad topics, it still can find reasonable topics, demonstrating **anchor-beta**'s greater robustness to suboptimal M.

L_2 (Sometimes) Improves Generalization As Figure 2 shows, **anchor-**L_2 sometimes improves held-out development likelihood for the smaller

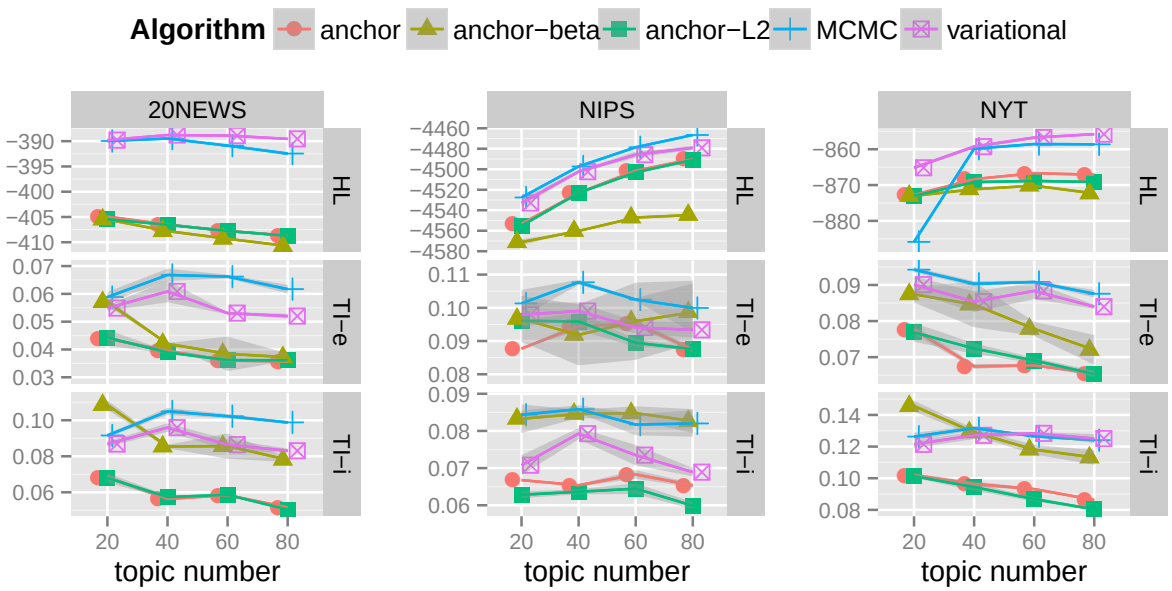


Figure 3: Comparing **anchor-beta** and **anchor-**L_2 against the original **anchor** and the traditional **variational** and **MCMC** on HL score and TI score. **variational** and **mcmc** provide the best held-out generalization. **anchor-beta** sometimes gives the best TI score and is consistently better than **anchor**. The specialized vocabulary of NIPS causes high variance for the extrinsic interpretability evaluation (TI-e).

Topic	Shared Words	Original (Top, green) vs. Informed L_2 (Bottom, orange)
soviet	american make president **soviet** union **war** years	gorbachev moscow russian force economic world europe political communist lead reform germany country **military** state **service** washington **bush** **army** unite **chief troops officer** **nuclear** time week
district	**assembly** board city **county district member** state york	representative manhattan brooklyn queens election bronx council island local incumbent housing municipal **people party group social** **republican** year make years **friend** **vote** **compromise** million
peace	american force government israel **peace** political president state unite washington	war military country minister leaders nation world palestinian israeli election **offer justice aid deserve** make **bush** years **fair** **clinton** **hand**
arms	**arms** bush congress force iraq make north nuclear president state washington weapon	administration treaty missile defense war military korea reagan **agree agreement** american **accept** unite **share** **clinton** years
trade	**administration** america american country **economic** government make president state **trade** unite washington	world market japan foreign china policy price political **business economy** **congress** year years **clinton** **bush** **buy**

Table 3: Examples of topic comparison between **anchor** and informed **anchor-**L_2. A topic is labeled with the anchor word for that topic. The **bold** words are the informed prior from LIWC. With an informed prior, relevant words appear in the top words of a topic; this also draws in other related terms (red).

20NEWS corpus. However, the λ selected on development data does not always improve test set performance. This, in Figure 3, **anchor-beta** closely tracks **anchor**. Thus, L_2 regularization does not hurt generalization while imparting expressiveness and robustness to parameter settings.

Beta Improves Interpretability Figure 3 shows that **anchor-beta** improves topic interpretability (TI) compared to unregularized anchor methods. In this section, we try to understand why.

We first compare the topics from the original **anchor** against **anchor-beta** to analyze the topics qualitatively. Table 5 shows that **beta** regularization promotes rarer words within a topic and demotes common words. For example, in the topic about hockey with the anchor word game, "run" and "good"—ambiguous, polysemous words—in the unregularized topic are replaced by "playoff"

Topic	anchor	anchor-beta
frequently	article write don doe make time people good file question	article write don doe make people time good email file
debate	write article people make don doe god key government time	people make god article write don doe key point government
wings	game team write wings article win red play hockey year	game team wings win red hockey play season player fan
stats	player team write game article stats year good play doe	stats player season league baseball fan team individual playoff nhl
compile	program file write email doe windows call problem run don	compile program code file ftp advance package error windows sun

Table 4: Topics from **anchor** and **anchor-beta** with $M = 100$ on 20NEWS with 20 topics. Each topic is identified with its associated anchor word. When $M = 100$, the topics of **anchor** suffer: the four colored words appear in almost every topic. **anchor-beta**, in contrast, is less sensitive to suboptimal M.

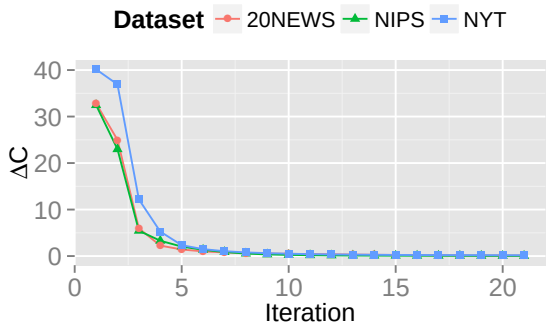


Figure 4: Convergence of anchor coefficient C for **anchor-beta**. ΔC is the difference of current C from the C at the previous iteration. C is converged within ten iterations for all three datasets.

and "trade" in the regularized topic. These words are less ambiguous and more likely to make sense to a consumer of topic models.

Figure 5 shows why this happens. Compared to the unregularized topics from **anchor**, the beta regularized topic steals from the rich and creates a more uniform distribution. Thus, highly frequent words do not as easily climb to the top of the distribution, and the topics reflect topical, relevant words rather than globally frequent terms.

6 Conclusion

A topic model is a popular tool for quickly getting the gist of large corpora. However, running such an analysis on these large corpora entail a substantial computational cost. While techniques such as **anchor** algorithms offer faster solutions, it comes at the cost of the expressive priors common in Bayesian formulations.

This paper introduces two different regularizations that offer users more interpretable models and the ability to inject prior knowledge without sacrificing the speed and generalizability of the underlying approach. However, one sacrifice that this approach does make is the beautiful theoretical guarantees of previous work. An important piece of future work is a theoretical understanding of generalizability in extensible, regularized models.

Incorporating other regularizations could further improve performance or unlock new applications. Our regularizations do not explicitly encourage sparsity; applying other regularizations such as L_1 could encourage true sparsity (Tibshirani, 1994), and structured priors (Andrzejewski et al., 2009) could efficiently incorporate constraints on topic models.

These regularizations could improve spectral algorithms for latent variables models, improving the performance for other NLP tasks such as latent variable PCFGs (Cohen et al., 2013) and HMMs (Anandkumar et al., 2012), combining the flexibility and robustness offered by priors with the speed and accuracy of new, scalable algorithms.

Acknowledgments

We would like to thank the anonymous reviewers, Hal Daumé III, Ke Wu, and Ke Zhai for their helpful comments. This work was supported by NSF Grant IIS-1320538. Boyd-Graber is also supported by NSF Grant CCF-1018625. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsor.

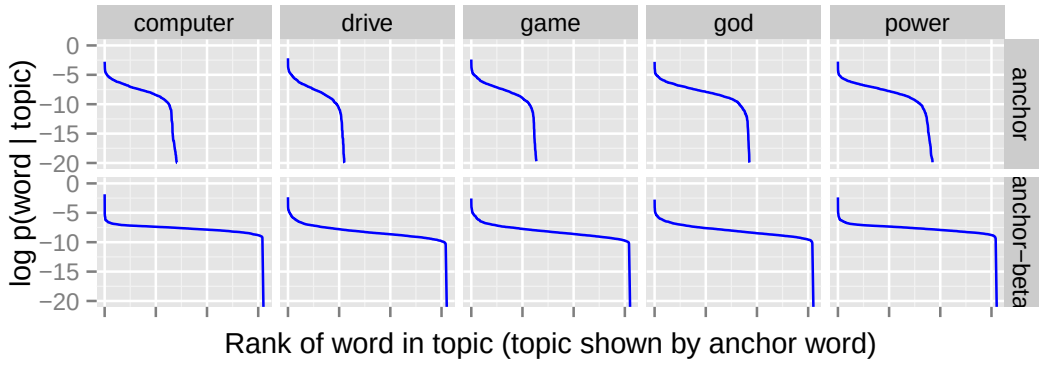


Figure 5: How beta regularization influences the topic distribution. Each topic is identified with its associated anchor word. Compared to the unregularized **anchor** method, **anchor-beta** steals probability mass from the "rich" and prefers a smoother distribution of probability mass. These words often tend to be unimportant, polysemous words common across topics.

Topic	Shared Words	**anchor** (Top, green) vs. **anchor-beta** (Bottom, orange)
computer	computer means science screen	system phone university problem doe work windows internet software chip mac set fax technology information data
		quote mhz pro processor ship remote print devices complex cpu electrical transfer ray engineering serial reduce
power	power play period supply ground light battery engine	car good make high problem work back turn control current small time
		circuit oil wire unit water heat hot ranger input total joe plug
god	god jesus christian bible faith church life christ belief religion hell word lord truth love	people make things true doe
		sin christianity atheist peace heaven
game	game team player play win fan hockey season baseball red wings score division league goal leaf cup toronto	run good
		playoff trade
drive	drive disk hard scsi controller card floppy ide mac bus speed monitor switch apple cable internal port meg	problem work
		ram pin

Table 5: Comparing topics—labeled by their anchor word—from **anchor** and **anchor-beta**. With beta regularization, relevant words are promoted, while more general words are suppressed, improving topic coherence.

References

Animashree Anandkumar, Daniel Hsu, and Sham M. Kakade. 2012. A method of moments for mixture models and hidden markov models. In *Proceedings of Conference on Learning Theory*.

David Andrzejewski, Xiaojin Zhu, and Mark Craven. 2009. Incorporating domain knowledge into topic modeling via Dirichlet forest priors. In *Proceedings of the International Conference of Machine Learning*.

Sanjeev Arora, Rong Ge, Yoni Halpern, David M. Mimno, Ankur Moitra, David Sontag, Yichen Wu, and Michael Zhu. 2012a. A practical algorithm for topic modeling with provable guarantees. *CoRR*, abs/1212.4777.

Sanjeev Arora, Rong Ge, and Ankur Moitra. 2012b. Learning topic models - going beyond svd. *CoRR*, abs/1204.1956.

David M. Blei and John D. Lafferty. 2005. Correlated topic models. In *Proceedings of Advances in Neural Information Processing Systems*.

David M. Blei, Andrew Ng, and Michael Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3.

Samuel Brody and Mirella Lapata. 2009. Bayesian word sense induction. In *Proceedings of the European Chapter of the Association for Computational Linguistics*, Athens, Greece.

Jonathan Chang, Jordan Boyd-Graber, Chong Wang, Sean Gerrish, and David M. Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Proceedings of Advances in Neural Information Processing Systems*.

Shay B. Cohen and Noah A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Shay Cohen, Karl Stratos, Michael Collins, Dean P. Foster, and Lyle Ungar. 2013. Experiments with spectral learning of latent-variable PCFGs. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the Association for Computational Linguistics*.

David Donoho and Victoria Stodden. 2003. When does non-negative matrix factorization give correct decomposition into parts? page 2004. MIT Press.

Miroslav Dudík, Steven J. Phillips, and Robert E. Schapire. 2004. Performance guarantees for regularized maximum entropy density estimation. In *Proceedings of Conference on Learning Theory*.

Vladimir Eidelman, Jordan Boyd-Graber, and Philip Resnik. 2012. Topic models for dynamic translation model adaptation. In *Proceedings of the Association for Computational Linguistics*.

Jenny Rose Finkel and Christopher D. Manning. 2009. Hierarchical bayesian domain adaptation. In *Conference of the North American Chapter of the Association for Computational Linguistics*, Morristown, NJ, USA.

Mark Galassi, Jim Davies, James Theiler, Brian Gough, Gerard Jungman, Michael Booth, and Fabrice Rossi. 2003. *Gnu Scientific Library: Reference Manual*. Network Theory Ltd.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl 1):5228–5235.

David Hall, Daniel Jurafsky, and Christopher D. Manning. 2008. Studying the history of ideas using topic models. In *Proceedings of Emperical Methods in Natural Language Processing*.

Yuening Hu, Jordan Boyd-Graber, Brianna Satinoff, and Alison Smith. 2013. Interactive topic modeling. *Machine Learning Journal*.

Matt L. Jockers. 2013. *Macroanalysis: Digital Methods and Literary History*. Topics in the Digital Humanities. University of Illinois Press.

Ravindran Kannan, Hadi Salmasian, and Santosh Vempala. 2005. The spectral method for general mixture models. In *Proceedings of Conference on Learning Theory*.

Ken Lang. 2007. 20 newsgroups data set.

Jey Han Lau, David Newman, and Timothy Baldwin. 2014. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In *Proceedings of the European Chapter of the Association for Computational Linguistics*.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://www.cs.umass.edu/ mccallum/mallet.

Thomas P. Minka. 2000. Estimating a dirichlet distribution. Technical report, Microsoft. http://research.microsoft.com/en-us/um/people/minka/papers/dirichlet/.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic evaluation of topic coherence. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Andrew Y. Ng. 2004. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the International Conference of Machine Learning*.

Michael Paul and Roxana Girju. 2010. A two-dimensional topic-aspect model for discovering multi-faceted topics. In *Association for the Advancement of Artificial Intelligence*.

James W. Pennebaker and Martha E. Francis. 1999. *Linguistic Inquiry and Word Count*. Lawrence Erlbaum, 1 edition, August.

Lawrence R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Jason Rennie. 2003. On l2-norm regularization and the Gaussian prior.

Sam Roweis. 2002. NIPS 1-12 Dataset.

Evan Sandhaus. 2008. The New York Times annotated corpus. http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2008T19.

Jayaram Sethuraman. 1994. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4:639–650.

Robert Tibshirani. 1994. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288.

Stefan Wager, Sida Wang, and Percy Liang. 2013. Dropout training as adaptive regularization. In *Proceedings of Advances in Neural Information Processing Systems*, pages 351–359.

Hanna Wallach, David Mimno, and Andrew McCallum. 2009. Rethinking LDA: Why priors matter. In *Proceedings of Advances in Neural Information Processing Systems*.

Limin Yao, David Mimno, and Andrew McCallum. 2009. Efficient methods for topic model inference on streaming document collections. In *Knowledge Discovery and Data Mining*.

Ke Zhai, Jordan Boyd-Graber, Nima Asadi, and Mohamad Alkhouja. 2012. Mr. LDA: A flexible large scale topic modeling package using variational inference in mapreduce. In *Proceedings of World Wide Web Conference*.

A Bayesian Mixed Effects Model of Literary Character

David Bamman
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
dbamman@cs.cmu.edu

Ted Underwood
Department of English
University of Illinois
Urbana, IL 61801, USA
tunder@illinois.edu

Noah A. Smith
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
nasmith@cs.cmu.edu

Abstract

We consider the problem of automatically inferring latent character types in a collection of 15,099 English novels published between 1700 and 1899. Unlike prior work in which character types are assumed responsible for probabilistically generating *all* text associated with a character, we introduce a model that employs multiple effects to account for the influence of extra-linguistic information (such as author). In an empirical evaluation, we find that this method leads to improved agreement with the preregistered judgments of a literary scholar, complementing the results of alternative models.

1 Introduction

Recent work in NLP has begun to exploit the potential of entity-centric modeling for a variety of tasks: Chambers (2013) places entities at the center of probabilistic frame induction, showing gains over a comparable event-centric model (Cheung et al., 2013); Bamman et al. (2013) explicitly learn character types (or "personas") in a dataset of Wikipedia movie plot summaries; and entity-centric models form one dominant approach in coreference resolution (Durrett et al., 2013; Haghighi and Klein, 2010).

One commonality among all of these very different probabilistic approaches is that each learns statistical regularities about how entities are depicted in text (whether for the sake of learning a set of semantic roles, character types, or linking anaphora to the entities to which they refer). In each case, the text we observe associated with an entity in a document is directly dependent on the class of entity—and only that class. This relationship between entity and text is a theoretical assumption, with important consequences for learning: entity types learned in this way will be increasingly similar the more similar the domain, author, and other extra-linguistic effects are between them.[1] While in many cases the topically similar types learned under this assumption may be desirable, we explore here the alternative, in which entity types are learned in a way that controls for such effects. In introducing a model based on different assumptions, we provide a method that complements past work and provides researchers with more flexible tools to infer different kinds of character types.

We focus here on the literary domain, exploring a large collection of 15,099 English novels published in the 18th and 19th centuries. By accounting for the influence of individual authors while inferring latent character types, we are able to learn personas that cut across different authors more effectively than if we learned types conditioned on the text alone. Modeling the language used to describe a character as the joint result of that character's latent type and of other formal variables allows us to test multiple models of character and assess their value for different interpretive problems. As a test case, we focus on separating character from authorial diction, but this approach can readily be generalized to produce models that provisionally distinguish character from other factors (such as period, genre, or point of view) as well.

2 Literary Background

Inferring character is challenging from a literary perspective partly because scholars have not reached consensus about the meaning of the term. It may seem obvious that a "character" is a representation of a (real or imagined) person, and many humanists do use the term that way. But there is

[1] For example, many entities in Early Modern English texts may be judged to be more similar to each other than to entities from later texts simply by virtue of using *hath* and other archaic verb forms.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 370–379,
Baltimore, Maryland, USA, June 23-25 2014.

an equally strong critical tradition that treats character as a formal dimension of narrative. To describe a character as a "blocking figure" or "first-person narrator," for instance, is a statement less about the attributes of an imagined person than about a narrative function (Keen, 2003). Characters are in one sense collections of psychological or moral attributes, but in another sense "word-masses" (Forster, 1927). This tension between "referential" and "formalist" models of character has been a centrally "divisive question in ... literary theory" (Woloch, 2003).

Considering primary source texts (as distinct from plot summaries) forces us to confront new theoretical questions about character. In a plot summary (such as those explored by Bamman et al., 2013), a human reader may already have used implicit models of character to extract high-level features. To infer character types from raw narrative text, researchers need to explicitly model the relationship of character to narrative form. This is not a solved problem, even for human readers.

For instance, it has frequently been remarked that the characters of Charles Dickens share certain similarities—including a reliance on tag phrases and recurring tics. A referential model of character might try to distinguish this common stylistic element from underlying "personalities." A strictly formalist model might refuse to separate authorial diction from character at all. In practice, human readers can adopt either perspective: we recognize that characters have a "Dickensian" quality but also recognize that a Dickens villain is (in one sense) more like villains in other authors than like a Dickensian philanthropist. Our goal is to show that computational methods can support the same range of perspectives—allowing a provisional, flexible separation between the referential and formal dimensions of narrative.

3 Data

The dataset for this work consists of 15,099 distinct narratives drawn from HathiTrust Digital Library.[2] From an initial collection of 469,200 volumes written in English and published between 1700 and 1899 (including poetry, drama, and nonfiction as well as prose narrative), we extract 32,209 volumes of prose fiction, remove duplicates and fuse multi-volume works to create the final dataset. Since the original texts were produced by scanning and running OCR on physical books, we automatically correct common OCR errors and trim front and back matter from the volumes using the page-level classifiers and HMM of Underwood et al. (2013)

Many aspects of this process would be simpler if we used manually-corrected texts, such as those drawn from Project Gutenberg. But we hope to produce research that has historical as well as computational significance, and doing so depends on the provenance of a collection. Gutenberg's decentralized selection process tends to produce exceptionally good coverage of currently-popular genres like science fiction, whereas HathiTrust aggregates university libraries. Library collections are not guaranteed to represent the past perfectly, but they are larger, and less strongly shaped by contemporary preferences.

The goal of this work is to provide a method to infer a set of character types in an unsupervised fashion from the data. As with prior work (Bamman et al., 2013), we define this target, a character *persona*, as a distribution over several categories of typed dependency relations:[3]

1. agent: the actions of which a character is the agent (i.e., verbs for which the character holds an nsubj or agent relation).
2. patient: the actions of which a character is the patient (i.e., verbs for which the character holds a dobj or nsubjpass relation).
3. possessive: the objects that a character possesses (i.e., all words for which the character holds a poss relation).
4. predicative: attributes predicated of a character (i.e., adjectives or nouns holding an nsubj relation to the character, with an inflection of *be* as a child).

This set captures the constellation of what a character *does* and *has done to them*, what they *possess*, and what they are described as *being*.

While previous work uses the Stanford CoreNLP toolkit to identify characters and extract typed dependencies for them, we found this approach to be too slow for the scale of our data (a total of 1.8 billion tokens); in particular, syntactic parsing, with cubic complexity in sentence length, and out-of-the-box coreference resolution (with thousands of potential antecedents) prove to be

[2] http://www.hathitrust.org

[3] All categories are described using the Stanford typed dependencies (de Marneffe and Manning, 2008), but any syntactic formalism is equally applicable.

the biggest bottlenecks.

Before addressing character inference, we present here a prerequisite NLP pipeline that scales well to book-length documents.[4] This pipeline uses the Stanford POS tagger (Toutanova et al., 2003), the linear-time MaltParser (Nivre et al., 2007) for dependency parsing (trained on Stanford typed dependencies), and the Stanford named entity recognizer (Finkel et al., 2005). It includes the following components for clustering character name mentions, resolving pronominal coreference, and reducing vocabulary dimensionality.

3.1 Character Clustering

First, let us terminologically distinguish between a character *mention* in a text (e.g., the token *Tom* on page 141 of *The Adventures of Tom Sawyer*) and a character *entity* (e.g., TOM SAWYER the character, to which that token refers). To resolve the former to the latter, we largely follow Davis et al. (2003) and Elson et al. (2010): we define a set of initial characters corresponding to each unique character name that is not a subset of another (e.g., *Mr. Tom Sawyer*) and deterministically create a set of allowable variants for each one (*Mr. Tom Sawyer* → *Tom*, *Sawyer*, *Tom Sawyer*, *Mr. Sawyer*, and *Mr. Tom*); then, from the beginning of the book to the end, we greedily assign each mention to the most recently linked entity for whom it is a variant. The result constitutes our set of characters, with all mentions partitioned among them.

3.2 Pronominal Coreference Resolution

While the character clustering stage is essentially performing proper noun coreference resolution, approximately 74% of references to characters in books come in the form of pronouns.[5] To resolve this more difficult class at the scale of an entire book, we train a log-linear discriminative classifier only on the task of resolving *pronominal* anaphora (i.e., ignoring generic noun phrases such as *the paint* or *the rascal*).

For this task, we annotated a set of 832 coreference links in 3 books (*Pride and Prejudice*, *The Turn of the Screw*, and *Heart of Darkness*) and featurized coreference/antecedent pairs with:

1. The syntactic dependency path from a pronoun to its potential antecedent (e.g., dobj↑pred→↓pred↓nsubj (where → denotes movement across sentence boundaries).
2. The salience of the antecedent character (defined as the count of that character's named mentions in the previous 500 words).
3. The antecedent part of speech.
4. Whether or not the pronoun and antecedent appear in the same quotation scope (false if one appears in a quotation and one outside).
5. Whether or not the two agree for gender.
6. The syntactic tree distance between the two.
7. The linear (word) distance between the two.

With this featurization and training data, we train a binary logistic regression classifier with ℓ_1 regularization (where negative examples are comprised of all character entities in the previous 100 words not labeled as the true antecedent). In a 10-fold cross-validation on predicting the true nearest antecedent for a pronominal anaphor, this method achieves an average accuracy of 82.7%.

With this trained model, we then select the highest-scoring antecedent within 100 words for each pronominal anaphor in our data.

3.3 Dimensionality Reduction

To manage the degrees of freedom in the model described in §4, we perform dimensionality reduction on the vocabulary by learning word embeddings with a log-linear continuous skip-gram language model (Mikolov et al., 2013) on the entire collection of 15,099 books. This method learns a low-dimensional real-valued vector representation of each word to predict all of the words in a window around it; empirically, we find that with a sufficient window size (we use $n = 10$), these word embeddings capture semantic similarity (placing topically similar words near each other in vector space).[6] We learn a 100-dimensional embedding for each of the 512,344 words in our vocabulary.

To create a partition over the vocabulary, we use hard K-means clustering (with Euclidean distance) to group the 512,344 word types into 1,000 clusters. We then agglomeratively cluster those 1,000 groups to assign bitstring representations to each one, forming a balanced binary tree by only merging existing clusters at equal levels in the hi-

[4] All code is available at `http://www.ark.cs.cmu.edu/literaryCharacter`

[5] Over all 15,099 narratives, the average number of character proper name mentions is 1,673; the average number of gendered singular pronouns (*he, she, him, his, her*) is 4,641.

[6] In comparison, Brown et al. (1992) clusters learned from the same data capture *syntactic* similarity (placing functionally similar words in the same cluster).

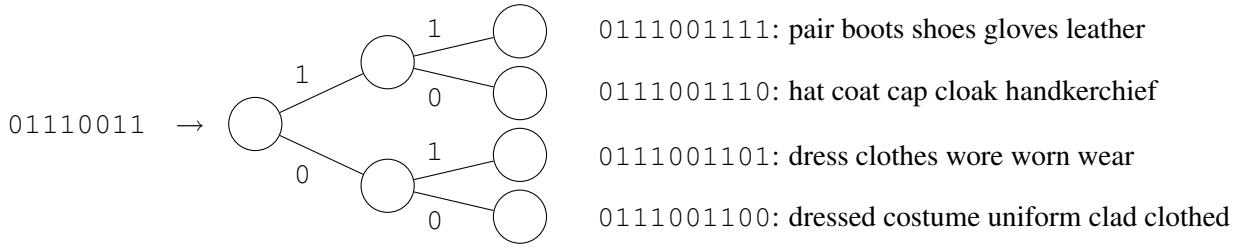


Figure 1: Bitstring representations of neural agglomerative clusters, illustrating the leaf nodes in a binary tree rooted in the prefix `01110011`. Bitstring encodings of intermediate nodes and terminal leaves result by following the left (`0`) and right (`1`) branches of the merge tree created through agglomerative clustering.

erarchy. We use Euclidean distance as a fundamental metric and a group-average similarity function for calculating the distance between groups. Fig. 1 illustrates four of the 1,000 learned clusters.

4 Model

In order to separate out the effects that a character's persona has on the words that are associated with them (as opposed to other factors, such as time period, genre, or author), we adopt a hierarchical Bayesian approach in which the words we observe are generated conditional on a combination of different effects captured in a log-linear (or "maximum entropy") distribution.

Maximum entropy approaches to language modeling have been used since Rosenfeld (1996) to incorporate long-distance information, such as previously-mentioned trigger words, into n-gram language models. This work has since been extended to a Bayesian setting by applying both a Gaussian prior (Chen and Rosenfeld, 2000), which dampens the impact of any individual feature, and sparsity-inducing priors (Kazama and Tsujii, 2003; Goodman, 2004), which can drive many feature weights to 0. The latter have been applied specifically to the problem of estimating word probabilities with sparse additive generative (SAGE) models (Eisenstein et al., 2011), where sparse extra-linguistic effects can influence a word probability in a larger generative setting.

In contrast to previous work in which the probability of a word linked to a character is dependent entirely on the character's latent persona, in our model, we see the probability of a word as dependent on: (i) the **background** likelihood of the word, (ii) the **author**, so that a word becomes more probable if a particular author tends to use it more, and (iii) the character's **persona**, so that a word is more probable if appearing with a particular persona. Intuitively, if the author *Jane Austen* is associated with a high weight for the word *manners*, and all personas have little effect for this word, then *manners* will have little impact on deciding which persona a particular Austen character embodies, since its presence is explained largely by Austen having penned the word. While we address only the author as an observed effect, this model is easily extended to other features as well, including period, genre, point of view, and others.

The generative story runs as follows (Figure 2 depicts the full graphical model): Let there be M unique authors in the data, P latent personas (a hyperparameter to be set), and V words in the vocabulary (in the general setting these may be word types; in our data the vocabulary is the set of 1,000 unique cluster IDs). Each role type $r \in \{\textsf{agent}, \textsf{patient}, \textsf{possessive}, \textsf{predicative}\}$ and vocabulary word v (here, a cluster ID) is associated with a real-valued vector $\eta_{r,v} = [\eta^{meta}_{r,v}, \eta^{pers}_{r,v}, \eta^{0}_{r,v}]$ of length $M + P + 1$. The first $M + P$ elements are drawn from a Laplace prior with mean $\mu = 0$ and scale $\lambda = 1$; the last element $\eta^{0}_{r,v}$ is an unregularized bias term accounting for the background. Each element in this vector captures the log-additive effect of each author, persona, and the background distribution on the word's probability (Eq. 1, below).

Much like latent Dirichlet allocation (Blei et al., 2003), each document d in our dataset draws a multinomial distribution θ_d over personas from a shared Dirichlet prior α, which captures the proportion of each character type in that particular document. Every character c in the document draws its persona p from this document-specific multinomial. Given document metadata m (here, one of a set of M authors) and persona p, each tuple of a role r with word w is assumed to be drawn from Eq. 1 in Fig. 3. This SAGE model can be understood as a log-linear distribution with three kinds of features (metadata, persona, and back-

$$P(w \mid m, p, r, \eta) = \exp\left(\eta^{meta}_{r,w}[m] + \eta^{pers}_{r,w}[p] + \eta^{0}_{r,w}\right) \Big/ \sum_{v=1}^{V} \exp\left(\eta^{meta}_{r,v}[m] + \eta^{pers}_{r,v}[p] + \eta^{0}_{r,v}\right) \quad (1)$$

$$P(b \mid m, p, r, \eta) = \prod_{j=0}^{n-1} \begin{cases} \text{logit}^{-1}\left(\eta^{meta}_{r,b_{1:j}}[m] + \eta^{pers}_{r,b_{1:j}}[p] + \eta^{0}_{r,b_{1:j}}\right) & \text{if } b_{j+1} = 1 \\ 1 - \text{logit}^{-1}\left(\eta^{meta}_{r,b_{1:j}}[m] + \eta^{pers}_{r,b_{1:j}}[p] + \eta^{0}_{r,b_{1:j}}\right) & \text{otherwise} \end{cases} \quad (2)$$

Figure 3: Parameterizations of the SAGE word distribution. Eq. 1 is a "flat" multinomial logistic regression with one η-vector per role and word. Eq. 2 uses the hierarchical softmax formulation, with one η-vector per role and node in the binary tree of word clusters, giving a distribution over bit strings (b) with the same number of parameters as Eq. 1.

ground bias).

4.1 Hierarchical Softmax

The partition function in Eq. 1 can lead to slow inference for any reasonably-sized vocabulary. To address this, we reparameterize the model by exploiting the structure of the agglomerative clustering in §3.3 to perform a hierarchical softmax, following Goodman (2001), Morin and Bengio (2005) and Mikolov et al. (2013).

The bitstring representations by which we encode each word in the vocabulary serve as natural, and inherently meaningful, intermediate classes that correspond to semantically related subsets of the vocabulary, with each bitstring prefix denoting one such class. Longer bitstrings correspond to more fine-grained classes. In the example shown in Figure 1, `011100111` is one such intermediate class, containing the union of *pair, boots, shoes, gloves leather* and *hat, coat, cap cloak, handkerchief*. Because these classes recursively partition the vocabulary, they offer a convenient way to reparameterize the model through the chain rule of probability.

Consider, for example, a word represented as the bitstring $c = \texttt{01011}$; calculating $P(c = \texttt{01011})$—we suppress conditioning variables for clarity—involves the product: $P(c_1 = \texttt{0}) \times P(c_2 = \texttt{1} \mid c_1 = \texttt{0}) \times P(c_3 = \texttt{0} \mid c_{1:2} = \texttt{01}) \times P(c_4 = \texttt{1} \mid c_{1:3} = \texttt{010}) \times P(c_5 = \texttt{1} \mid c_{1:4} = \texttt{0101})$.

Since each multiplicand involves a binary prediction, we can avoid partition functions and use the classic binary logistic regression.[7] We have converted the V-way multiclass logistic regression problem of Eq. 1 into a sequence of $\log V$ evaluations (assuming a perfectly balanced tree). Given m, p, and r (as above) we let $b = b_1 b_2 \cdots b_n$ denote the bitstring representation of a word cluster, and the distribution is given by Eq. 2 in Fig. 3.

In this paramaterization, rather than one η-vector for each role and vocabulary term, we have one η-vector for each role and conditional binary decision in the tree (each bitstring prefix). Since the tree is binary with V leaves, this yields the same total number of parameters. As Goodman (2001) points out, while this reparameterization is exact for true probabilities, it remains an approximation for estimated models (with generalization behavior dependent on how well the class hierarchy is supported by the data). In addition to enabling faster inference, one advantage of the bitstring representation and the hierarchical softmax parameterization is that we can easily calculate probabilities of clusters at different granularities.

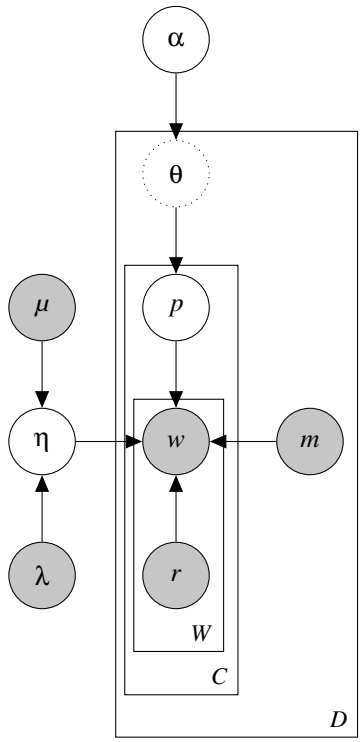
4.2 Inference

Our primary quantities of interest in this model are p (the personas for each character) and η, the effects that each author and persona have on the probability of a word. Rather than adopting a fully Bayesian approach (e.g., sampling all variables), we infer these values using stochastic EM, alternating between collapsed Gibbs sampling for each p and maximizing with respect to η.

Collapsed Gibbs for personas.[8] At each step, the required quantity is the probability that character c in document d has persona z, given everything else. This is proportional to the number of other characters in document d who also (currently) have that persona (plus the Dirichlet hyperparameter which acts as a smoother) times the probability (under $p_{d,c} = z$) of all of the words

[7] Recall that logistic regression lets $P_{LR}(y = 1 \mid x, \beta) = \text{logit}^{-1}(x^\top \beta) = 1/(1 + \exp -x^\top \beta)$ for binary dependent variable y, independent variables x, and coefficients β.

[8] We assume the reader is familiar with collapsed Gibbs sampling as used in latent-variable NLP models.



P	Number of personas (hyperparameter)
D	Number of documents
C_d	Number of characters in document d
$W_{d,c}$	Number of (cluster, role) tuples for character c
m_d	Metadata for document d (ranges over M authors)
θ_d	Document d's distribution over personas
$p_{d,c}$	Character c's persona
j	An index for a $\langle r, w \rangle$ tuple in the data
w_j	Word cluster ID for tuple j
r_j	Role for tuple $j \in \{\textsf{agent}, \textsf{patient}, \textsf{poss}, \textsf{pred}\}$
η	Coefficients for the log-linear language model
μ, λ	Laplace mean and scale (for regularizing η)
α	Dirichlet concentration parameter

Figure 2: **Above:** Probabilistic graphical model. Observed variables are shaded, latent variables are clear, and collapsed variables are dotted. **Below:** Definition of variables.

observed in each role r for that character:

$$(count(z; p_{d,-c}) + \alpha_z) \times \prod_{r=1}^{R} \prod_{j:r_j=r} P(b_j \mid m, p, r, \eta) \quad (3)$$

The metadata features (like author, etc.) influence this probability by being constant for all choices of z; e.g., if the coefficient learned for *Austen* for vocabulary term *manners* is high and all coefficients for all z are close to zero, then the probability of *manners* will change little under different choices of z. Eq. 3 contains one multiplicand for every word associated with a character, and only one term reflecting the influence of the shared document multinomial. The implication is that, for major characters with many observed words, the words will dominate the choice of persona; where the document influence would have a bigger effect is with characters for whom we don't have much data. In that case, it can act as a kind of informed background; given what little data we have for that character, it would nudge us toward the character types that the other characters in the book embody.

Given an assignment of all p, we choose η to maximize the conditional log-likelihood of the words, as represented by their bitstring cluster IDs, given the observed author and background effects and the sampled personas. This equates to solving $4V$ ℓ_1-regularized logistic regressions (see Eq. 2 in Figure 3), one for each role type and bitstring prefix, each with $M + P + 1$ parameters. We apply OWL-QN (Andrew and Gao, 2007) to minimize the ℓ_1-regularized objective with an absolute convergence threshold of 10^{-5}.

5 Evaluation

While standard NLP and machine learning practice is to evaluate the performance of an algorithm on a held-out gold standard, articulating what a true "persona" might be for a character is inherently problematic. Rather, we evaluate the performance and output of our model by preregistering a set of 29 hypotheses of varying scope and difficulty and comparing the performance of different models in either confirming, or failing to confirm, those hypotheses. This kind of evaluation was previously applied to a subjective text measurement problem by Sim et al. (2013).

All hypotheses were created by a literary scholar with specialization in the period to not only give an empirical measure of the strengths and weaknesses of different models, but also to help explore exactly what the different models may, or may not, be learning. All preregistered hypotheses establish the degrees of similarity among three characters, taking the form: "character X is more similar to character Y than either X or Y is to a distractor character Z"; for a given model and definition of distance under that model, each hypothesis yields two yes/no decisions that we can evaluate:

- $distance(X, Y) < distance(X, Z)$
- $distance(X, Y) < distance(Y, Z)$

To tease apart the different kinds of similarities we hope to explore, we divide the hypotheses into four classes:

A. This class constitutes **sanity checks**: character X and Y are more similar to each other in every way than to character Z. E.g.: Elizabeth Bennet in *Pride and Prejudice* resembles Elinor Dashwood in *Sense and Sensibility* (Jane Austen) more than either character resembles Allen Quatermain in *Allen Quatermain* (H. Rider Haggard). (Austenian protagonists should resemble each other more than they resemble a grizzled hunter.)
B. This class captures our ability to identify two characters in the same author as being more similar to each other than to a closely related character in a **different** author. E.g.: Wickham in *Pride and Prejudice* resembles Willoughby in *Sense and Sensibility* (Jane Austen) more than either character resembles Mr. Rochester in *Jane Eyre* (Charlotte Brontë).
C. This class captures our ability to discriminate among similar characters in the **same** author. In these hypotheses, two characters X and Y from the same author are more similar to each other than to a third character Z from that same author. E.g.: Wickham in *Pride and Prejudice* (Jane Austen) resembles Willoughby in *Sense and Sensibility* more than either character resembles Mr. Darcy in *Pride and Prejudice.*
D. This class constitutes more difficult, **exploratory** hypotheses, including differences among point of view. E.g.: Montoni in *Mysteries of Udolpho* (Radcliffe) resembles Heathcliff in *Wuthering Heights* (Emily Brontë) more than either resembles Mr. Bennet in *Pride and Prejudice.* (Testing our model's ability to discern similarities in spite of elapsed time.)

All 29 hypotheses can be found in a supplementary technical report (Bamman et al., 2014). We emphasize that the full set of hypotheses was locked *before* the model was estimated.

6 Experiments

Part of the motivation of our mixed effects model is to be able to tackle hypothesis class C—by factoring out the influence of a particular author on the learning of personas, we would like to be able to discriminate between characters that all have a common authorial voice. In contrast, the Persona Regression model of Bamman et al. (2013), which uses metadata variables (like authorship) to encourage entities with similar covariates to have similar personas, reflects an assumption that makes it likely to perform well at class B.

To judge their respective strengths on different hypothesis classes, we evaluate three models:

1. The mixed-effects Author/Persona model (described above), which includes author information as a metadata effect; here, each η-vector (of length $M + P + 1$) contains a parameter for each of the distinct authors in our data, a parameter for each persona, and a background parameter.
2. A Basic persona model, which ablates author information but retains the same log-linear architecture; here, the η-vector is of size $P + 1$ and does not model author effects.
3. The Persona Regression model of Bamman et al. (2013).

All models are run with $P \in \{10, 25, 50, 100, 250\}$ personas; Persona Regression additionally uses $K = 25$ latent topics. All configurations use the full dataset of 15,099 novels, and all characters with at least 25 total roles (a total of 257,298 entities). All experiments are run with 50 iterations of Gibbs sampling to collect samples for the personas p, alternating with maximization steps for η. The value of α is optimized using slice sampling (with a non-informative prior) every 5 iterations. The value of λ is held constant at 1. At the end of inference, we calculate the posterior distributions over personas for all characters as the sampling probability of the final iteration.

To formally evaluate "similarity" between two characters, we measure the Jensen-Shannon divergence between personas (calculated as the average JS distance over the cluster distributions for each role type), marginalizing over the characters' posterior distributions over personas; two characters with a lower JS divergence are judged to be more similar than two characters with a higher one.

As a Baseline, we also evaluate all hypotheses on a model with no latent variables whatsoever, which instead measures similarity as the average JS divergence between the empirical word distributions over each role type.

Table 1 presents the results of this comparison; for all models with latent variables, we report the average of 5 sampling runs with different random initializations. Figure 4 provides a syn-

P	Model	Hypothesis Class A	B	C	D
250	Author/Persona	1.00	0.58	0.75	0.42
	Basic Persona	1.00	0.73	0.58	0.53
	Persona Reg.	0.90	0.70	0.58	0.44
100	Author/Persona	0.98	0.68	0.70	0.46
	Basic Persona	0.95	0.73	0.53	0.47
	Persona Reg.	0.93	0.78	0.63	0.49
50	Author/Persona	0.95	0.73	0.63	0.50
	Basic Persona	0.98	0.75	0.48	0.53
	Persona Reg.	1.00	0.75	0.65	0.38
25	Author/Persona	1.00	0.63	0.65	0.50
	Basic Persona	1.00	0.63	0.50	0.50
	Persona Reg.	0.90	0.78	0.60	0.39
10	Author/Persona	0.95	0.63	0.70	0.51
	Basic Persona	0.78	0.80	0.48	0.46
	Persona Reg.	0.90	0.73	0.43	0.41
Baseline		1.00	0.63	0.58	0.37

Table 1: Agreement rates with preregistered hypotheses, averaged over 5 sampling runs with different initializations.

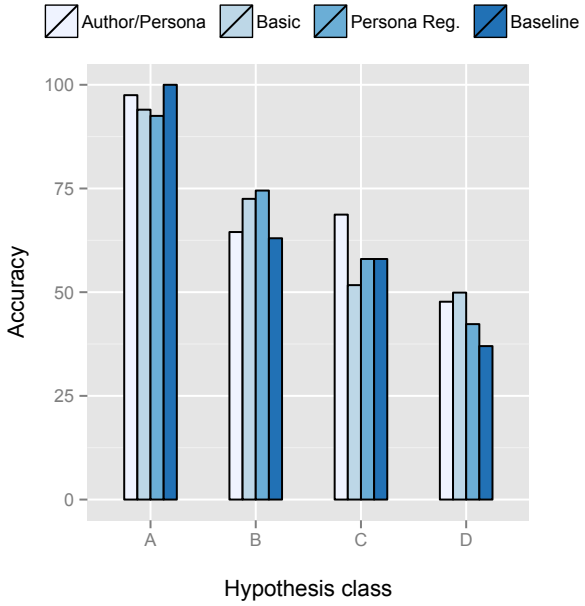


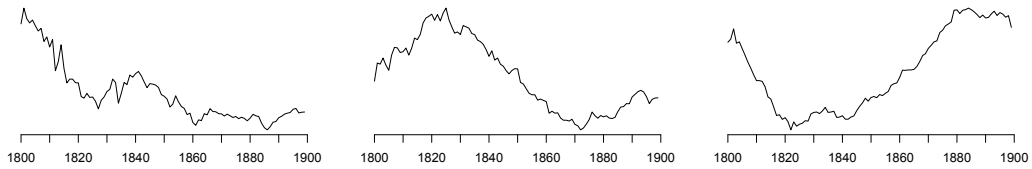
Figure 4: Synopsis of table 1: average accuracy across all P. Persona regression is best able to judge characters in one author to be more similar to each other than to characters in another (B), while our mixed-effects Author/Persona model outperforms other models at discriminating characters in the same author (C).

opsis of this table by illustrating the average accuracy across all choice of P. All models, including the baseline, perform well on the sanity checks (A). As expected, the Persona Regression model performs best at hypothesis class B (correctly judging two characters from the same author to be more similar to each other than to a character from a different author); this behavior is encouraged in this model by allowing an author (as an external metadata variable) to directly influence the persona choice, which has the effect of pushing characters from the same author to embody the same character type. Our mixed effects Author/Persona model, in contrast, outperforms the other models at hypothesis class C (correctly discriminating different character types present in the same author). By discounting author-specific lexical effects during persona inference, we are better able to detect variation among the characters of a single author that we are not able to capture otherwise. While these different models complement each other in this manner, we note that there is no absolute separation among them, which may be suggestive of the degree to which the formal and referential dimensions are fused in novels. Nevertheless, the strengths of these different models on these different hypothesis classes gives us flexible alternatives to use depending on the kinds of character types we are looking to infer.

7 Analysis

The latent personas inferred from this model will support further exploratory analysis of literary history. Figure 2 illustrates this with a selection of three character types learned, displaying characteristic clusters for all role types, along with the distribution of that persona's use across time and the gender distribution of characters embodying that persona. In general, the personas learned so far do not align neatly with character types known to literary historians. But they do have legible associations both with literary genres and with social categories. Even though gender is not an observable variable known to the model during inference, personas tend to be clearly gendered. This is not in itself surprising (since literary scholars know that assumptions about character are strongly gendered), but it does suggest that diachronic analysis of latent character types might cast new light on the history of gender in fiction. This is especially true since the distribution of personas across the time axis similarly reveals coherent trends.

Table 3 likewise illustrates what our model learns by presenting a sample of the fixed effects learned for a set of five major 19th-century authors. These are clusters that are conditionally more likely to appear associated with a character in a work by the given author than they are in the overall data; by factoring this information out of the inference process for learning character types (by attributing its presence in a text to the author



Agent	carried ran threw rose fell suddenly is seems	sent received arrived appeared struck showed returned immediately waiting	turns begins returns thinks loves calls does knows comes
Patient	wounded killed murdered suffer yield acknowledge free saved unknown	wounded killed murdered destroy bind crush attend haste proceed	thinks loves calls love hope true turn hold show
Poss	death happiness future lips cheek brow mouth fingers tongue	army officers troops soldiers band armed party join camp	lips cheek brow eyes face eye table bed chair
Pred	crime guilty murder youth lover hers dead living died	king emperor throne general officer guard soldier knight hero	beautiful fair fine good kind ill dead living died
% Female	12.2	3.7	54.7

Table 2: Snapshots of three personas learned from the $P = 50$, Author/Persona model. Gender and time proportions are calculated by summing and normalizing the posterior distributions over all characters with that feature. We truncate time series at 1800 due to data sparsity before that date; the y-axis illustrates the frequency of its use in a given year, relative to its lifetime.

Author	clusters
Jane Austen	praise gift consolation letter read write character natural taste
Charlotte Brontë	lips cheek brow book paper books hat coat cap
Charles Dickens	hat coat cap table bed chair hand head hands
Herman Melville	boat ship board hat coat cap feet ground foot
Jules Verne	journey travel voyage master company presence success plan progress

Table 3: Characteristic possessive clusters in a sample of major 19th-century authors.

rather than the persona), we are able to learn personas that cut across different topics more effectively than if a character type is responsible for explaining the presence of these terms as well.

8 Conclusion

Our method establishes the possibility of representing the relationship between character and narrative form in a hierarchical Bayesian model. Postulating an interaction between authorial diction and character allows models that consider the effect of the author to more closely reproduce a human reader's judgments, especially by learning to distinguish different character types within a single author's oeuvre. This opens the door to considering other structural and formal dimensions of narration. For instance, representation of character is notoriously complicated by narrative point of view (Booth, 1961); and indeed, comparisons between first-person narrators and other characters are a primary source of error for all models tested above. The strategy we have demonstrated suggests that it might be productive to address this by modeling the interaction of character and point of view as a separate effect analogous to authorship.

It is also worth noting that the models tested above diverge from many structuralist theories of narrative (Propp, 1998) by allowing multiple instances of the same persona in a single work. Learning structural limitations on the number of "protagonists" likely to coexist in a single story, for example, may be another fruitful area to explore. In all cases, the machinery of hierarchical models gives us the flexibility to incorporate such effects at will, while also being explicit about the theoretical assumptions that attend them.

9 Acknowledgments

We thank the reviewers for their helpful comments. The research reported here was supported by a National Endowment for the Humanities start-up grant to T.U., U.S. National Science Foundation grant CAREER IIS-1054319 to N.A.S., and an ARCS scholarship to D.B. This work was made possible through the use of computing resources made available by the Pittsburgh Supercomputing Center. Eleanor Courtemanche provided advice about the history of narrative theory.

References

Galen Andrew and Jianfeng Gao. 2007. Scalable training of l_1-regularized log-linear models. In *Proc. of ICML*.

David Bamman, Brendan O'Connor, and Noah A. Smith. 2013. Learning latent personas of film characters. *Proc. of ACL*.

David Bamman, Ted Underwood, and Noah A. Smith. 2014. Appendix to 'A Bayesian mixed effects model of literary character'. Technical report, Carnegie Mellon University, University of Illinois-Urbana Champaign.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Wayne Booth. 1961. *The Rhetoric of Fiction*. University of Chicago Press, Chicago.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Nathanael Chambers. 2013. Event schema induction with a probabilistic entity-driven model. In *Proc. of EMNLP*, Seattle, Washington, USA.

Stanley F. Chen and Roni Rosenfeld. 2000. A survey of smoothing techniques for me models. *IEEE Transactions on Speech and Audio Processing*, 8(1):37–50.

Jackie Chi Kit Cheung, Hoifung Poon, and Lucy Vanderwende. 2013. Probabilistic frame induction. In *Proc. of NAACL*.

Peter T. Davis, David K. Elson, and Judith L. Klavans. 2003. Methods for precise named entity matching in digital collections. In *Proc. of JCDL*, Washington, DC, USA.

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

Greg Durrett, David Hall, and Dan Klein. 2013. Decentralized entity-level modeling for coreference resolution. In *Proc. of ACL*.

Jacob Eisenstein, Amr Ahmed, and Eric P. Xing. 2011. Sparse additive generative models of text. In *Proc. of ICML*.

David K. Elson, Nicholas Dames, and Kathleen R. McKeown. 2010. Extracting social networks from literary fiction. In *Proc. of ACL*, Stroudsburg, PA, USA.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proc. of ACL*.

E. M. Forster. 1927. *Aspects of the Novel*. Harcourt, Brace & Co.

Joshua Goodman. 2001. Classes for fast maximum entropy training. In *Proc. of ICASSP*.

Joshua Goodman. 2004. Exponential priors for maximum entropy models. In *Proc. of NAACL*.

Aria Haghighi and Dan Klein. 2010. Coreference resolution in a modular, entity-centered model. In *Proc. of NAACL*.

Jun'ichi Kazama and Jun'ichi Tsujii. 2003. Evaluation and extension of maximum entropy models with inequality constraints. In *Proc. of EMNLP*.

Suzanne Keen. 2003. *Narrative Form.* Palgrave Macmillan, Basingstoke.

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

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proc. of AISTATS*.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13:95–135, 5.

Vladimir Propp. 1998. *Morphology of the Folktale*. University of Texas Press, 2nd edition.

Roni Rosenfeld. 1996. A maximum entropy approach to adaptive statistical language modelling. *Computer Speech and Language*, 10(3):187 – 228.

Yanchuan Sim, Brice D. L. Acree, Justin H. Gross, and Noah A. Smith. 2013. Measuring ideological proportions in political speeches. In *Proc. of EMNLP*, Seattle, Washington, USA.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. of NAACL*.

Ted Underwood, Michael L Black, Loretta Auvil, and Boris Capitanu. 2013. Mapping mutable genres in structurally complex volumes. In *Proc. of IEEE International Conference on Big Data*.

Alex Woloch. 2003. *The One vs. the Many: Minor Characters and the Space of the Protagonist in the Novel*. Princeton University Press, Princeton NJ.

Collective Tweet Wikification based on Semi-supervised Graph Regularization

Hongzhao Huang[1], Yunbo Cao[2], Xiaojiang Huang[2], Heng Ji[1], Chin-Yew Lin[2]
[1]Computer Science Department, Rensselaer Polytechnic Institute, Troy, NY 12180, USA
[2]Microsoft Research Asia, Beijing 100080, P.R.China
{huangh9,jih}@rpi.edu[1],
{yunbo.cao,xiaojih,cyl}@microsoft.com[2]

Abstract

Wikification for tweets aims to automatically identify each concept mention in a tweet and link it to a concept referent in a knowledge base (e.g., Wikipedia). Due to the shortness of a tweet, a collective inference model incorporating global evidence from multiple mentions and concepts is more appropriate than a non-collecitve approach which links each mention at a time. In addition, it is challenging to generate sufficient high quality labeled data for supervised models with low cost. To tackle these challenges, we propose a novel semi-supervised graph regularization model to incorporate both local and global evidence from multiple tweets through three fine-grained relations. In order to identify semantically-related mentions for collective inference, we detect meta path-based semantic relations through social networks. Compared to the state-of-the-art supervised model trained from 100% labeled data, our proposed approach achieves comparable performance with 31% labeled data and obtains 5% absolute F1 gain with 50% labeled data.

1 Introduction

With millions of tweets posted daily, Twitter enables both individuals and organizations to disseminate information, from current affairs to breaking news in a timely fashion. In this work, we study the wikification (Disambiguation to Wikipedia) task (Mihalcea and Csomai, 2007) for tweets, which aims to automatically identify each *concept mention* in a tweet, and link it to a *concept* referent in a knowledge base (KB) (e.g., Wikipedia). For example, as shown in Figure 1, *Hawks* is an identified mention, and its correct referent concept in Wikipedia is *Atlanta Hawks*. An end-to-end wikification system needs to solve two sub-problems: (i) concept mention detection, (ii) concept mention disambiguation.

Wikification is a particularly useful task for short messages such as tweets because it allows a reader to easily grasp the related topics and enriched information from the KB. From a system-to-system perspective, wikification has demonstrated its usefulness in a variety of applications, including coreference resolution (Ratinov and Roth, 2012) and classification (Vitale et al., 2012).

Sufficient labeled data is crucial for supervised models. However, manual wikification annotation for short documents is challenging and time-consuming (Cassidy et al., 2012). The challenges are: (i) *unlinkability*, a valid concept may not exist in the KB. (ii) *ambiguity*, it is impossible to determine the correct concept due to the dearth of information within a single tweet or multiple correct answer. For instance, it would be difficult to determine the correct referent concept for "*Gators*" in t_1 in Figure 1. Linking "*UCONN*" in t_3 to *University of Connecticut* may also be acceptable since *Connecticut Huskies* is the athletic team of the university. (iii) *prominence*, it is challenging to select a set of linkable mentions that are important and relevant. It is not tricky to select "*Fans*", "*slump*", and "*Hawks*" as linkable mentions, but other mentions such as "*stay up*" and "*stay positive*" are not prominent. Therefore, it is challenging to create sufficient high quality labeled tweets for supervised models and worth considering semi-supervised learning with the exploration of unlabeled data.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 380–390,
Baltimore, Maryland, USA, June 23-25 2014.

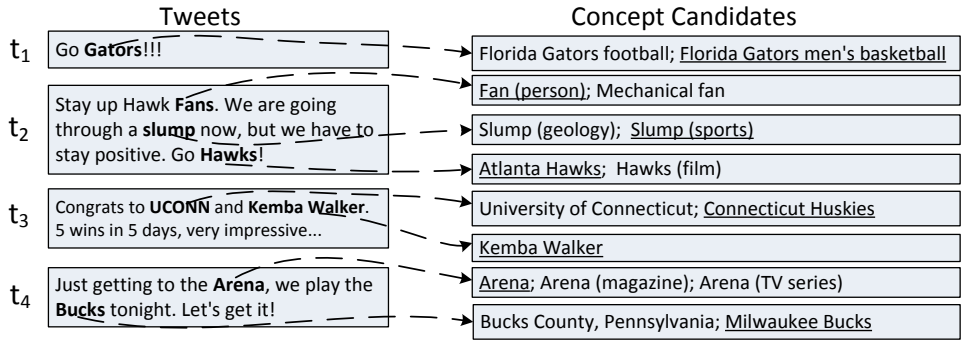


Figure 1: An illustration of Wikification Task for Tweets. Concept mentions detected in tweets are marked as bold, and correctly linked concepts are underlined. The concept candidates are ranked by their prior popularity which will be explained in section 4.1, and only top 2 ranked concepts are listed.

However, when selecting semi-supervised learning frameworks, we noticed another unique challenge that tweets pose to wikification due to their informal writing style, shortness and noisiness. The context of a single tweet usually cannot provide enough information for prominent mention detection and similarity computing for disambiguation. Therefore, a collective inference model over multiple tweets in the semi-supervised setting is desirable. For instance, the four tweets in Figure 1 are posted by the same author within a short time period. If we perform collective inference over them we can reliably link ambiguous mentions such as "*Gators*", "*Hawks*", and "*Bucks*" to basketball teams instead of other concepts such as the county *Bucks County*.

In order to address these unique challenges for wikification for the short tweets, we employ graph-based semi-supervised learning algorithms (Zhu et al., 2003; Smola and Kondor, 2003; Blum et al., 2004; Zhou et al., 2004; Talukdar and Crammer, 2009) for collective inference by exploiting the manifold (cluster) structure in both unlabeled and labeled data. These approaches normally assume label smoothness over a defined graph, where the nodes represent a set of labeled and unlabeled instances, and the weighted edges reflect the closeness of each pair of instances. In order to construct a semantic-rich graph capturing the similarity between mentions and concepts for the model, we introduce three novel fine-grained relations based on a set of local features, social networks and meta paths.

The main contributions of this paper are summarized as follows:

- To the best of our knowledge, this is the first effort to explore graph-based semi-supervised learning algorithms for the wikification task.
- We propose a novel semi-supervised graph regularization model performing collective inference for joint mention detection and disambiguation. Our approach takes advantage of three proposed principles to incorporate both local and global evidence from multiple tweets.
- We propose a meta path-based unified framework to detect both explicitly and implicitly relevant mentions.

2 Preliminaries

Concept and Concept Mention We define a *concept* c as a Wikipedia article (e.g., *Atlanta Hawks*), and a *concept mention* m as an n-gram from a specific tweet. Each concept has a set of textual representation fields (Meij et al., 2012), including *title* (the title of the article), *sentence* (the first sentence of the article), *paragraph* (the first paragraph of the article), *content* (the entire content of the article), and *anchor* (the set of all anchor texts with incoming links to the article).

Wikipedia Lexicon Construction We first construct an offline lexicon with each entry as $\langle m, \{c_1, ..., c_k\}\rangle$, where $\{c_1, ..., c_k\}$ is the set of possible referent concepts for the mention m. Following the previous work (Bunescu, 2006; Cucerzan, 2007; Hachey et al., 2013), we extract the possible mentions for a given concept c using the following resources: the title of c; the aliases appearing in the introduction and infoboxes of c (e.g., *The Evergreen State* is an alias of *Washington state*); the titles of pages redirecting to c (e.g., *State of Washington* is a redirecting page of *Washington (state)*); the titles of the disambigua-

tion pages containing c; and all the anchor texts appearing in at least 5 pages with hyperlinks to c (e.g., *WA* is a mention for the concept *Washington (state)* in the text "*401 5th Ave N [[Seattle]], [[Washington (state)—WA]] 98109 USA*". We also propose three heuristic rules to extract mentions (i.e., different combinations of the family name and given name for a person, the headquarters of an organization, and the city name for a sports team).

Concept Mention Extraction Based on the constructed lexicon, we then consider all n-grams of size $\leq n$ (n=7 in this paper) as concept mention candidates if their entries in the lexicon are not empty. We first segment @usernames and #hashtags into regular tokens (e.g., @amandapalmer is segmented as *amanda palmer* and #WorldWaterDay is split as *World Water Day*) using the approach proposed by (Wang et al., 2011). Segmentation assists finding concept candidates for these non-regular mentions.

3 Principles and Approach Overview

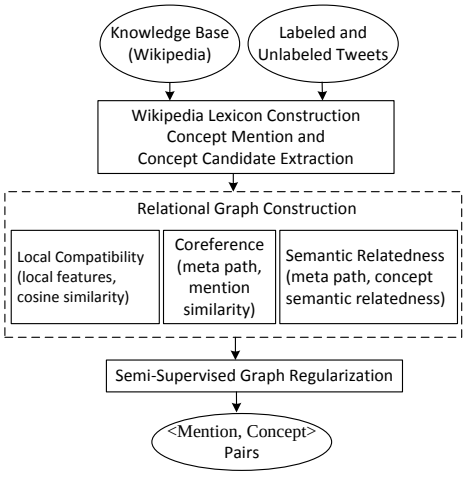


Figure 2: Approach Overview.

3.1 Principles

A single tweet may not provide enough evidence to identify prominent mentions and infer their correct referent concepts due to the lack of contextual information. To tackle this problem, we propose to incorporate global evidence from multiple tweets and performing collective inference for both mention identification and disambiguation. We first introduce the following three principles that our approach relies on.

Principle 1 (Local compatibility): *Two pairs of $\langle m, c\rangle$ with strong local compatibility tend to have similar labels.* Mentions and their correct referent concepts usually tend to share a set of characteristics such as string similarity between m and c (e.g., $\langle$*Chicago*, *Chicago*$\rangle$ and $\langle$*Facebook*, *Facebook*$\rangle$). We define the *local compatibility* to model such set of characteristics.

Principle 2 (Coreference): *Two coreferential mentions should be linked to the same concept.* For example, if we know "*nc*" and "*North Carolina*" are coreferential, then they should both be linked to *North Carolina*.

Principle 3 (Semantic Relatedness): *Two highly semantically-related mentions are more likely to be linked to two highly semantically-related concepts.* For instance, when "*Sweet 16*" and "*Hawks*" often appear together within relevant contexts, they can be reliably linked to two baseketball-related concepts *NCAA Men's Division I Basketball Championship* and *Atlanta Hawks*, respectively.

3.2 Approach Overview

Given a set of tweets $\langle t_1, ..., t_{|T|}\rangle$, our system first generates a set of candidate concept mentions, and then extracts a set of candidate concept referents for each mention based on the Wikipedia lexicon. Given a pair of mention and its candidate referent concept $\langle m, c\rangle$, the remaining task of wikification is to assign either a positive label if m should be selected as a prominently linkable mention and c is its correct referent concept, or otherwise a negative label. The label assignment is obtained by our semi-supervised graph regularization framework based on a relational graph, which is constructed from *local compatibility*, *coreference*, and *semantic relatedness* relations. The overview of our approach is as illustrated in Figure 2.

4 Relational Graph Construction

We first construct the relational graph $G = \langle V, E\rangle$, where $V = \{v_1, ..., v_n\}$ is a set of nodes and $E = \{e_1, ..., e_m\}$ is a set of edges. Each $v_i = \langle m_i, c_i\rangle$ represents a tuple of mention m_i and its referent concept candidate c_i. An edge is added between two nodes v_i and v_j if there is a proposed relation based on the three principles described in section 3.1.

4.1 Local Compatibility

We first compute local compatibility (Principle 1) by considering a set of novel local features to cap-

ture the importance and relevance of a mention m to a tweet t, as well as the correctness of its linkage to a concept c. We have designed a number of features which are similar to those commonly used in wikification and entity linking work (Meij et al., 2012; Guo et al., 2013; Mihalcea and Csomai, 2007).

Mention Features We define the following features based on information from mentions.

- $IDF_f(m) = \log(\frac{|C|}{df(m)})$, where $|C|$ is the total number of concepts in Wikipedia and $df(m)$ is the total number of concepts in which m occurs, and f indicates the field property, including *title*, *content*, and *anchor*.
- $Keyphraseness(m) = \frac{|C_a(m)|}{df(m)}$ to measure how likely m is used as an anchor in Wikipedia, where $C_a(m)$ is the set of concepts where m appears as an anchor.
- $LinkProb(m) = \frac{\sum_{c \in C_a(m)} count(m,c)}{\sum_{c \in C} count(m,c)}$, where $count(m, c)$ indicates the number of occurrence of m in c.
- $SNIL(m)$ and $SNCL(m)$ to count the number of concepts that are equal to or contain a sub-n-gram of m, respectively (Meij et al., 2012).

Concept Features The concept features are solely based on Wikipedia, including the number of incoming and outgoing links for c, and the number of words and characters in c.

Mention + Concept Features This set of features considers information from both mentions and concepts:

- **prior popularity** $prior(m, c) = \frac{count(m,c)}{\sum_{c'} count(m,c')}$, where $count(m, c)$ measures the frequency of the anchor links from m to c in Wikipedia.
- $TF_f(m, c) = \frac{count_f(m,c)}{|f|}$ to measure the relative frequency of m in each field representation f of c, normalized by the length of f. The fields include *title*, *sentence*, *paragraph*, *content* and *anchor*.
- $NCT(m, c)$, $TCN(m, c)$, and $TEN(m, c)$ to measure whether m contains the title of c, whether the title of c contains m, and whether m equals to the title of c, respectively.

Context Features This set of features include (i) Context Capitalization features, which indicate whether the current mention, the token before, and the token after are capitalized. (ii) tf-idf based features, which include the dot product of two word vectors v_c and v_t, and the average tf-idf value of common items in v_c and v_t, where v_c and v_t are the top 100 tf-idf word vectors in c and t.

Local Compatibility Computation For each node $v_i = \langle m_i, c_i \rangle$, we collect its local features as a feature vector $F_i = \langle f_1, f_2, ..., f_d \rangle$. To avoid features with large numerical values that dominate other features, the value of each feature is re-scaled using feature standardization approach. The cosine similarity is then adopted to compute the local compatibility of two nodes and construct a k *nearest neighbor* (kNN) graph, where each node is connected to its k nearest neighboring nodes. We compute the weight matrix that represents the local compatibility relation as:

$$W_{ij}^{loc} = \begin{cases} cosine(F_i, F_j) & j \in kNN(i) \\ 0 & \text{Otherwise} \end{cases}$$

4.2 Meta Path

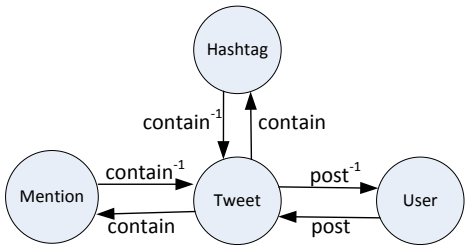


Figure 3: Schema of the Twitter network.

In this subsection, we introduce the concept meta path which will be used to detect coreference (section 4.3) and semantic relatedness relations (section 4.4).

A meta-path is a path defined over a network and composed of a sequence of relations between different object types (Sun et al., 2011b). In our experimental setting, we can construct a natural Twitter network summarized by the network schema in Figure 3. The network contains four types of objects: *Mention* (M), *Tweet* (T), *User* (U), and *Hashtag* (H). Tweets and mentions are connected by links "contain" and "contained by" (denoted as "$contain^{-1}$"); and other linked relationships can be described similarly.

We then define the following five types of meta paths to connect two mentions as:

- "M - T - M",
- "M - T - U - T - M",
- "M - T - H - T - M",
- "M - T - U - T - M - T - H - T - M",
- "M - T - H - T - M - T - U - T - M".

Each meta path represents one particular semantic relation. For instance, the first three paths are *basic* ones expressing the explicit relations that two mentions are from the same tweet, posted by the same user, and share the same #hashtag, respectively. The last two paths are *concatenated* ones which are constructed by concatenating the first three simple paths to express the implicit relations that two mentions co-occur with a third mention sharing either the same authorship or #hashtag. Such complicated paths can be exploited to detect more semantically-related mentions from wider contexts. For example, the relational link between "*narita airport*" and "*Japan*" would be missed without using the path "*narita airport* - t_1 - u_1 - t_2 - *american* - t_3 - h_1 - t_4 - *Japan*" since they don't directly share any authorships or #hashtags.

4.3 Coreference

A coreference relation (Principle 2) usually occurs across multiple tweets due to the highly redundant information in Twitter. To ensure high precision, we propose a simple yet effective approach utilizing the rich social network relations in Twitter.

We consider two mentions m_i and m_j coreferential if m_i and m_j share the same surface form or one is an abbreviation of the other, and at least one meta path exists between m_i and m_j. Then we define the weight matrix representing the coreferential relation as:

$$W_{ij}^{coref} = \begin{cases} 1.0 & \text{if } m_i \text{ and } m_j \text{ are coreferential,} \\ & \text{and } c_i = c_j \\ 0 & \text{Otherwise} \end{cases}$$

4.4 Semantic Relatedness

Ensuring topical coherence (Principle 3) has been beneficial for wikification on formal texts (e.g., News) by linking a set of semantically-related mentions to a set of semantically-related concepts simultaneously (Han et al., 2011; Ratinov et al., 2011; Cheng and Roth, 2013). However, the shortness of a single tweet means that it may not provide enough topical clues. Therefore, it is important to extend this evidence to capture semantic relatedness information from multiple tweets.

We define the semantic relatedness score between two mentions as $SR(m_i, m_j) = 1.0$ if at least one meta path exists between m_i and m_j, otherwise $SR(m_i, m_j) = 0$. In order to compute the semantic relatedness of two concepts c_i and c_j, we adopt the approach proposed by (Milne and Witten, 2008a):

$$SR(c_i, c_j) = 1 - \frac{\log \max(|C_i|, |C_j|) - \log |C_i \cap C_j|}{\log(|C|) - \log \min(|C_i|, |C_j|)},$$

where $|C|$ is the total number of concepts in Wikipedia, and C_i and C_j are the set of concepts that have links to c_i and c_j, respectively.

Then we compute a weight matrix representing the semantic relatedness relation as:

$$W_{ij}^{rel} = \begin{cases} SR(N_i, N_j) & \text{if } SR(N_i, N_j) \geq \delta \\ 0 & \text{Otherwise} \end{cases}$$

where $SR(N_i, N_j) = SR(m_i, m_j) \times SR(c_i, c_j)$ and $\delta = 0.3$, which is optimized from a development set.

4.5 The Combined Relational Graph

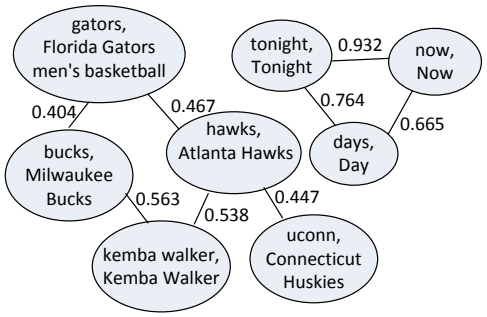


Figure 4: A example of the relational graph constructed for the example tweets in Figure 1. Each node represents a pair of $\langle m, c \rangle$, separated by a comma. The edge weight is obtained from the linear combination of the weights of the three proposed relations. Not all mentions are included due to the space limitations.

Based on the above three weight matrices W^{loc}, W^{coref}, and W^{rel}, we first obtain their corresponding transition matrices P^{loc}, P^{coref}, and P^{rel}, respectively. The entry P_{ij} of the transition matrix P for a weight matrix W is computed as $P_{ij} = \frac{W_{ij}}{\sum_k W_{ik}}$ such that $\sum_k P_{ik} = 1$. Then we obtain the combined graph G with weight matrix W, where $W_{ij} = \alpha P_{ij}^{loc} + \beta P_{ij}^{coref} + \gamma P_{ij}^{rel}$. α, β, and γ are three coefficients between 0 and 1 with the constraint that $\alpha + \beta + \gamma = 1$. They control the contributions of these three relations in our semi-supervised graph regularization model. We choose transition matrix to avoid the domination of one relation over others. An example graph of G is shown in Figure 4. Compared to the *referent graph* which considers each mention or concept as a node in previous graph-based re-ranking approaches (Han et al., 2011; Shen et al., 2013), our

novel graph representation has two advantages: (i) It can easily incorporate more features related to both mentions and concepts. (ii) It is more appropriate for our graph-based semi-supervised model since it is difficult to assign labels to a pair of mention and concept in the referent graph.

5 Semi-supervised Graph Regularization

Given the constructed relational graph with the weighted matrix W and the label vector Y of all nodes, we assume the first l nodes are labeled as Y_l and the remaining u nodes ($u = n - l$) are initialized with labels Y_u^0. Then our goal is to refine Y_u^0 and obtain the final label vector Y_u.

Intuitively, if two nodes are strongly connected, they tend to hold the same label. We propose a novel semi-supervised graph regularization framework based on the graph-based semi-supervised learning algorithm (Zhu et al., 2003):

$$\mathcal{Q}(\mathcal{Y}) = \mu \sum_{i=l+1}^{n} (y_i - y_i^0)^2 + \frac{1}{2} \sum_{i,j} W_{ij}(y_i - y_j)^2.$$

The first term is a loss function that incorporates the initial labels of unlabeled examples into the model. In our method, we adopt *prior popularity* (section 4.1) to initialize the labels of the unlabeled examples. The second term is a regularizer that smoothes the refined labels over the constructed graph. μ is a regularization parameter that controls the trade-off between initial labels and the consistency of labels on the graph. The goal of the proposed framework is to ensure that the refined labels of unlabeled nodes are consistent with their strongly connected nodes, as well as not too far away from their initial labels.

The above optimization problem can be solved directly since $\mathcal{Q}(\mathcal{Y})$ is convex (Zhu et al., 2003; Zhou et al., 2004). Let I be an identity matrix and D_W be a diagonal matrix with entries $D_{ii} = \sum_j W_{ij}$. We can split the weighted matrix W into four blocks as $W = \begin{bmatrix} W_{ll} & W_{lu} \\ W_{ul} & W_{uu} \end{bmatrix}$, where W_{mn} is an $m \times n$ matrix. D_w is split similarly. We assume that the vector of the labeled examples Y_l is fixed, so we only need to infer the refined label vector of the unlabeled examples Y_u. In order to minimize $\mathcal{Q}(\mathcal{Y})$, we need to find Y_u^* such that

$$\left.\frac{\partial Q}{\partial Y_u}\right|_{Y_u = Y_u^*} = (D_{uu} + \mu I_{uu})Y_u - W_{uu}Y_u - W_{ul}Y_l - \mu Y_u^0 = 0.$$

Therefore, a closed form solution can be derived as $Y_u^* = (D_{uu} + \mu I_{uu} - W_{uu})^{-1}(W_{ul}Y_l + \mu Y_u^0)$.

However, for practical application to a large-scale data set, an iterative solution would be more efficient to solve the optimization problem. Let Y_u^t be the refined labels after the t^{th} iteration, the iterative solution can be derived as:

$$Y_u^{t+1} = (D_{uu} + \mu I_{uu})^{-1}(W_{uu}Y_u^t + W_{ul}Y_l + \mu Y_u^0).$$

The iterative solution is more efficient since $(D_{uu} + \mu I_{uu})$ is a diagonal matrix and its inverse is very easy to compute.

6 Experiments

In this section we compare our approach with state-of-the-art methods as shown in Table 1.

6.1 Data and Scoring Metric

For our experiments we use a public data set (Meij et al., 2012) including 502 tweets posted by 28 verified users. The data set was annotated by two annotators. We randomly sample 102 tweets for development and the remaining for evaluation. We use a Wikipedia dump on May 3, 2013 as our knowledge base, which includes 30 million pages. For computational efficiency, we also filter some mention candidates by applying the preprocessing approach proposed in (Ferragina and Scaiella, 2010), and remove all the concepts with prior popularity less than 2% from an mention's concept set for each mention, similar to (Guo et al., 2013).

A mention and concept pair $\langle m, c \rangle$ is judged as correct if and only if m is linkable and c is the correct referent concept for m. To evaluate the performance of a wikification system, we use the standard precision, recall and F1 measures.

6.2 Experimental Results

The overall performance of various approaches is shown in Table 2. The results of the supervised method proposed by (Meij et al., 2012) are obtained from 5-fold cross validation. For our semi-supervised setting, we experimentally sample 200 tweets for training and use the remaining set as unlabeled and testing sets. In our semi-supervised regularization model, the matrix W^{loc} is constructed by a kNN graph ($k = 20$). The regularization parameter μ is empirically set to 0.1, and the coefficients α, β, and γ are learnt from the development set by considering all the combina-

Methods	Descriptions
TagMe	The same approach that is described in (Ferragina and Scaiella, 2010), which aims to annotate short texts based on prior popularity and semantic relatedness of concepts. It is basically an unsupervised approach, except that it needs a development set to tune the probability threshold for linkable mentions.
Meij	A state-of-the-art system described in (Meij et al., 2012), which is a supervised approach based on the random forest model. It performs mention detection and disambiguation jointly, and it is trained from 400 labeled tweets.
SSRegu$_1$	Our proposed model based on Principle 1, using 200 labeled tweets.
SSRegu$_{12}$	Our proposed model based on Principle 1 and 2, using 200 labeled tweets.
SSRegu$_{13}$	Our proposed model based on Principle 1 and 3, using 200 labeled tweets.
SSRegu$_{123}$	Our proposed full model based on Principle 1, 2 and 3, using 200 labeled tweets.

Table 1: Description of Methods.

Methods	Precision	Recall	F1
TagMe	0.329	0.423	0.370
Meij	0.393	0.598	0.475
SSRegu$_1$	0.538	0.435	0.481
SSRegu$_{12}$	0.638	0.438	0.520
SSRegu$_{13}$	0.541	0.457	0.495
SSRegu$_{123}$	0.650	0.441	**0.525**

Table 2: Overall Performance.

tions of values from 0 to 1 at 0.1 intervals[1]. In order to randomize the experiments and make the comparison fair, we conduct 20 test runs for each method and report the average scores across the 20 trials.

The relatively low performance of the baseline system *TagMe* demonstrates that only relying on prior popularity and topical information within a single tweet is not enough for an end-to-end wikification system for the short tweets. As an example, it is difficult to obtain topical clues in order to link the mention "*Clinton*" to *Hillary Rodham Clinton* by relying on the single tweet "*wolfblitzercnn: Behind the scenes on Clinton's Mideast trip #cnn*". Therefore, the system mistakenly links it to the most popular concept *Bill Clinton*.

In comparision with the supervised baseline proposed by (Meij et al., 2012), our model *SSRegu*$_1$ relying on *local compatibility* already achieves comparable performance with 50% of labeled data. This is because that our model performs collective inference by making use of the manifold (cluster) structure of both labeled and unlabeled data, and that the local compatibility relation is detected with high precision[2] (89.4%). For example, the following three pairs of mentions and concepts ⟨*pelosi, Nancy Pelosi*⟩, ⟨*obama, Barack Obama*⟩, and ⟨*gaddafi, Muammar Gaddafi*⟩ have strong local compatibility with each other since they share many similar characteristics captured by the local features such as string similarity between the mention and the concept. Suppose the first pair is labeled, then its positive label will be propagated to other unlabeled nodes through the local compatibility relation, and correctly predict the labels of other nodes.

Incorporating coreferential or semantic relatedness relation into *SSRegu*$_1$ provides further gains, demonstrating the effectiveness of these two relations. For instance, "*wh*" is correctly linked to *White House* by incorporating evidence from its coreferential mention "*white house*". The coreferential relation (Principle 2) is demonstrated to be more beneficial than the semantic relatedness relation (Principle 3) because the former is detected with much higher precision (99.7%) than the latter (65.4%).

Our full model *SSRegu*$_{123}$ achieves significant improvement over the supervised baseline (5% absolute F1 gain with 95.0% confidence level by the Wilcoxon Matched-Pairs Signed-Ranks Test), showing that incorporating global evidence from multiple tweets with fine-grained relations is beneficial. For instance, the supervised baseline fails to link "*UCONN*" and "*Bucks*" in our examples to *Connecticut Huskies* and *Milwaukee Bucks*, respectively. Our full model corrects these two wrong links by propagating evidence through the semantic links as shown in Figure 4 to obtain mutual ranking improvement. The best performance of our full model also illustrates that the three relations complement each other.

We also study the disambiguation performance for the annotated mentions, as shown in Table 3. We can easily see that our proposed approach using 50% labeled data achieves similar performance with the state-of-the-art supervised model with 100% labeled data. When the mentions are given, the unpervised approach *TagMe* has already

[1]These three coefficients are slightly different with different training data, a sample of them is: $\alpha = 0.4$, $\beta = 0.5$, and $\gamma = 0.1$

[2]Here we define precision as the percentage of links that holds the same label.

Methods	TagMe	Meij	SSRegu_{123}
Accuracy	0.710	0.779	0.772

Table 3: Disambiguation Performance.

Methods	Precision	Recall	F1
$\textbf{SSRegu}_{12}$	0.644	0.423	0.510
$\textbf{SSRegu}_{13}$	0.543	0.441	0.486
$\textbf{SSRegu}_{123}$	0.657	0.419	0.512

Table 4: The Performance of Systems Without Using Concatenated Meta Paths.

achieved reasonable performance. In fact, mention detection actually is the performance bottleneck of a tweet wikification system (Guo et al., 2013). Our system performs better in identifying the prominent mention.

6.3 Effect of Concatenated Meta Paths

In this work, we propose a unified framework utilizing meta path-based semantic relations to explore richer relevant context. Beyond the *basic* meta paths, we introduce *concatenated* ones by concatenating the basic ones. The performance of the system without using the concatenated meta paths is shown in Table 4. In comparison with the system based on all defined meta paths, we can clearly see that the systems using concatenated ones outperform those relying on the simple ones. This is because the concatenated meta paths can incorporate more relevant information with implicit relations into the models by increasing 1.6% coreference links and 9.3% semantic relatedness links. For example, the mention "*narita airport*" is correctly disambiguated to the concept "*Narita International Airport*" with higher confidence since its semantic relatedness relation with "*Japan*" is detected with the concatenated meta path as described in section 4.2.

6.4 Effect of Labeled Data Size

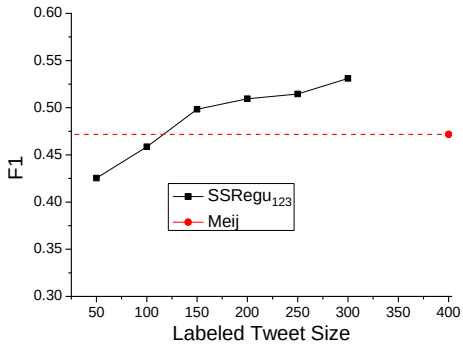


Figure 5: The effect of Labeled Tweet Size.

In previous experiments, we experimentally set the number of labeled tweets to be 200 for overall performance comparision with the baselines. In this subsection, we study the effect of labeled data size on our full model. We randomly sample 100 tweets as testing data, and randomly select 50, 100, 150, 200, 250, and 300 tweets as labeled data. 20 test runs are conducted and the average results are reported across the 20 trials, as shown in Figure 5. We find that as the size of the labeled data increases, our proposed model achieves better performance. It is encouraging to see that our approach, with only $\mathbf{31.3}\%$ labeled tweets (125 out of 400), already achieves a performance that is comparable to the state-of-the-art supervised model trained from 100% labeled tweets.

6.5 Parameter Analysis

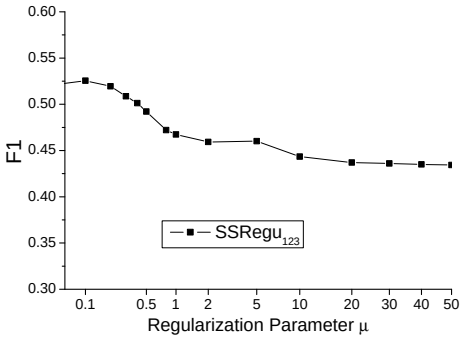


Figure 6: The effect of parameter μ.

In previous experiments, we empirically set the parameter $\mu = 0.1$. μ is the regularization parameter that controls the trade-off between initial labels and the consistency of labels on the graph. When μ increases, the model tends to trust more in the initial labels. Figure 6 shows the performance of our models by varying μ from 0.02 to 50. We can easily see that the system performce is stable when $\mu < 0.4$. However, when $\mu \geq 0.4$, the system performance dramatically decreases, showing that prior popularity is not enough for an end-to-end wikification system.

7 Related Work

The task of linking concept mentions to a knowledge base has received increased attentions over the past several years, from the linking of concept mentions in a single text (Mihalcea and Csomai, 2007; Milne and Witten, 2008b; Milne and Witten, 2008a; Kulkarni et al., 2009; He et al., 2011; Ratinov et al., 2011; Cassidy et al., 2012; Cheng and Roth, 2013), to the linking of a cluster of corefer-

ent named entity mentions spread throughout different documents (Entity Linking) (McNamee and Dang, 2009; Ji et al., 2010; Zhang et al., 2010; Ji et al., 2011; Zhang et al., 2011; Han and Sun, 2011; Han et al., 2011; Gottipati and Jiang, 2011; He et al., 2013; Li et al., 2013; Guo et al., 2013; Shen et al., 2013; Liu et al., 2013).

A significant portion of recent work considers the two sub-problems *mention detection* and *mention disambiguation* separately and focus on the latter by first defining candidate concepts for a deemed mention based on anchor links. Mention disambiguation is then formulated as a ranking problem, either by resolving one mention at each time (non-collective approaches), or by disambiguating a set of relevant mentions simultaneously (collective approaches). Non-collective methods usually rely on prior popularity and context similarity with supervised models (Mihalcea and Csomai, 2007; Milne and Witten, 2008b; Han and Sun, 2011), while collective approaches further leverage the global coherence between concepts normally through supervised or graph-based re-ranking models (Cucerzan, 2007; Milne and Witten, 2008b; Han and Zhao, 2009; Kulkarni et al., 2009; Pennacchiotti and Pantel, 2009; Ferragina and Scaiella, 2010; Fernandez et al., 2010; Radford et al., 2010; Cucerzan, 2011; Guo et al., 2011; Han and Sun, 2011; Han et al., 2011; Ratinov et al., 2011; Chen and Ji, 2011; Kozareva et al., 2011; Cassidy et al., 2012; Shen et al., 2013; Liu et al., 2013). Especially note that when applying the collective methods to short messages from social media, evidence from other messages usually needs to be considered (Cassidy et al., 2012; Shen et al., 2013; Liu et al., 2013). Our method is a collective approach with the following novel advancements: (i) A novel graph representation with fine-grained relations, (ii) A unified framework based on meta paths to explore richer relevant context, (iii) Joint identification and linking of mentions under semi-supervised setting.

Two most similar methods to ours were proposed by (Meij et al., 2012; Guo et al., 2013) by performing joint detection and disambiguation of mentions. (Meij et al., 2012) studied several supervised machine learning models, but without considering any global evidence either from a single tweet or other relevant tweets. (Guo et al., 2013) explored second order entity-to-entity relations but did not incorporate evidence from multiple tweets.

This work is also related to graph-based semi-supervised learning (Zhu et al., 2003; Smola and Kondor, 2003; Zhou et al., 2004; Talukdar and Crammer, 2009), which has been successfully applied in many Natural Language Processing tasks (Niu et al., 2005; Chen et al., 2006). We introduce a novel graph that incorporates three fine-grained relations. Our work is further related to meta path-based heterogeneous information network analysis (Sun et al., 2011b; Sun et al., 2011a; Kong et al., 2012; Huang et al., 2013), which has demonstrated advantages over homogeneous information network analysis without differentiating object types and relational links.

8 Conclusions

We have introduced a novel semi-supervised graph regularization framework for wikification to simultaneously tackle the unique challenges of annotation and information shortage in short tweets. To the best of our knowledge, this is the first work to explore the semi-supervised collective inference model to jointly perform mention detection and disambiguation. By studying three novel fine-grained relations, detecting semantically-related information with semantic meta paths, and exploiting the data manifolds in both unlabeled and labeled data for collective inference, our work can dramatically save annotation cost and achieve better performance, thus shed light on the challenging wikification task for tweets.

Acknowledgments

This work was supported by the U.S. Army Research Laboratory under Cooperative Agreement No. W911NF-09-2-0053 (NS-CTA), U.S. NSF CAREER Award under Grant IIS-0953149, U.S. DARPA Award No. FA8750-13-2-0041 in the Deep Exploration and Filtering of Text (DEFT) Program, IBM Faculty Award, Google Research Award and RPI faculty start-up grant. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

A. Blum, J. Lafferty, M. Rwebangira, and R. Reddy. 2004. Semi-supervised learning using randomized mincuts. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04.

Razvan Bunescu. 2006. Using encyclopedic knowledge for named entity disambiguation. In *EACL*, pages 9–16.

T. Cassidy, H. Ji, L. Ratinov, A. Zubiaga, and H. Huang. 2012. Analysis and enhancement of wikification for microblogs with context expansion. In *Proceedings of COLING 2012*.

Z. Chen and H. Ji. 2011. Collaborative ranking: A case study on entity linking. In *Proc. EMNLP2011*.

J. Chen, D. Ji, C Tan, and Z. Niu. 2006. Relation extraction using label propagation based semi-supervised learning. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*.

X. Cheng and D. Roth. 2013. Relational inference for wikification. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *EMNLP-CoNLL 2007*.

S. Cucerzan. 2011. Tac entity linking by performing full-document entity extraction and disambiguation. In *Proc. TAC 2011 Workshop*.

N. Fernandez, J. A. Fisteus, L. Sanchez, and E. Martin. 2010. Webtlab: A cooccurence-based approach to kbp 2010 entity-linking task. In *Proc. TAC 2010 Workshop*.

P. Ferragina and U. Scaiella. 2010. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10.

S. Gottipati and J. Jiang. 2011. Linking entities to a knowledge base with query expansion. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*.

Y. Guo, W. Che, T. Liu, and S. Li. 2011. A graph-based method for entity linking. In *Proc. IJCNLP2011*.

S. Guo, M. Chang, and E. Kiciman. 2013. To link or not to link? a study on end-to-end tweet entity linking. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

B. Hachey, W. Radford, J. Nothman, M. Honnibal, and J. Curran. 2013. Evaluating entity linking with wikipedia. *Artif. Intell.*

X. Han and L. Sun. 2011. A generative entity-mention model for linking entities with knowledge base. In *Proc. ACL2011*.

X. Han and J. Zhao. 2009. Named entity disambiguation by leveraging wikipedia semantic knowledge. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM 2009.

X. Han, L. Sun, and J. Zhao. 2011. Collective entity linking in web text: A graph-based method. In *Proc. SIGIR2011*.

J. He, M. de Rijke, M. Sevenster, R. van Ommering, and Y. Qian. 2011. Generating links to background knowledge: A case study using narrative radiology reports. In *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM.

Z. He, S. Liu, Y. Song, M. Li, M. Zhou, and H. Wang. 2013. Efficient collective entity linking with stacking. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

H. Huang, Z. Wen, D. Yu, H. Ji, Y. Sun, J. Han, and H. Li. 2013. Resolving entity morphs in censored data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

H. Ji, R. Grishman, H.T. Dang, K. Griffitt, and J. Ellis. 2010. Overview of the tac 2010 knowledge base population track. In *Text Analysis Conference (TAC) 2010*.

H. Ji, R. Grishman, and H.T. Dang. 2011. Overview of the tac 2011 knowledge base population track. In *Text Analysis Conference (TAC) 2011*.

X. Kong, P. Yu, Y. Ding, and J. Wild. 2012. Meta path-based collective classification in heterogeneous information networks. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12.

Z. Kozareva, K. Voevodski, and S. Teng. 2011. Class label enhancement via related instances. In *Proc. EMNLP2011*.

S. Kulkarni, A. Singh, G. Ramakrishnan, and S. Chakrabarti. 2009. Collective annotation of wikipedia entities in web text. In *KDD*.

Y. Li, C. Wang, F. Han, J. Han, D. Roth, and X. Yan. 2013. Mining evidences for named entity disambiguation. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13.

X. Liu, Y. Li, H. Wu, M. Zhou, F. Wei, and Y. Lu. 2013. Entity linking for tweets. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

P. McNamee and H.T. Dang. 2009. Overview of the tac 2009 knowledge base population track. In *Text Analysis Conference (TAC) 2009*.

E. Meij, W. Weerkamp, and M. de Rijke. 2012. Adding semantics to microblog posts. In *Proceedings of the fifth ACM international conference on Web search and data mining*, WSDM '12.

R. Mihalcea and A. Csomai. 2007. Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM '07.

D. Milne and I.H. Witten. 2008a. Learning to link with wikipedia. In *An effective, low-cost measure of semantic relatedness obtained from wikipedia links*. the Wikipedia and AI Workshop of AAAI.

D. Milne and I.H. Witten. 2008b. Learning to link with wikipedia. In *Proceeding of the 17th ACM conference on Information and knowledge management*, pages 509–518. ACM.

Z. Niu, D. Ji, and C. Tan. 2005. Word sense disambiguation using label propagation based semi-supervised learning. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*.

M. Pennacchiotti and P. Pantel. 2009. Entity extraction via ensemble semantics. In *Proc. EMNLP2009*.

W. Radford, B. Hachey, J. Nothman, M. Honnibal, and J. R. Curran. 2010. Cmcrc at tac10: Document-level entity linking with graph-based re-ranking. In *Proc. TAC 2010 Workshop*.

L. Ratinov and D. Roth. 2012. Learning-based multi-sieve co-reference resolution with knowledge. In *EMNLP*.

L. Ratinov, D. Roth, D. Downey, and M. Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*.

W. Shen, J. Wang, P. Luo, and M. Wang. 2013. Linking named entities in tweets with knowledge base via user interest modeling. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13.

A. Smola and R. Kondor. 2003. Kernels and regularization on graphs. *COLT*.

Y. Sun, R. Barber, M. Gupta, C. Aggarwal, and J. Han. 2011a. Co-author relationship prediction in heterogeneous bibliographic networks. In *Proceedings of the 2011 International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '11.

Y. Sun, J. Han, X. Yan, P. Yu, and T. Wu. 2011b. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *PVLDB*, 4(11).

P. Talukdar and K. Crammer. 2009. New regularized algorithms for transductive learning. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II*, ECML PKDD '09.

D. Vitale, P. Ferragina, and U. Scaiella. 2012. Classification of short texts by deploying topical annotations. In *ECIR*, pages 376–387.

K. Wang, C. Thrasher, and B. Hsu. 2011. Web scale nlp: A case study on url word breaking. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11.

W. Zhang, J. Su, C. Tan, and W. Wang. 2010. Entity linking leveraging automatically generated annotation. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*.

W. Zhang, J. Su, and C. L. Tan. 2011. A wikipedia-lda model for entity linking with batch size changing. In *Proc. IJCNLP2011*.

D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf. 2004. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16*.

X. Zhu, Z. Ghahramani, and J. Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*.

Zero-shot Entity Extraction from Web Pages

Panupong Pasupat
Computer Science Department
Stanford University
ppasupat@cs.stanford.edu

Percy Liang
Computer Science Department
Stanford University
pliang@cs.stanford.edu

Abstract

In order to extract entities of a fine-grained category from semi-structured data in web pages, existing information extraction systems rely on seed examples or redundancy across multiple web pages. In this paper, we consider a new zero-shot learning task of extracting entities specified by a natural language query (in place of seeds) given only a single web page. Our approach defines a log-linear model over latent extraction predicates, which select lists of entities from the web page. The main challenge is to define features on widely varying candidate entity lists. We tackle this by abstracting list elements and using aggregate statistics to define features. Finally, we created a new dataset of diverse queries and web pages, and show that our system achieves significantly better accuracy than a natural baseline.

1 Introduction

We consider the task of extracting entities of a given category (e.g., *hiking trails*) from web pages. Previous approaches either (i) assume that the same entities appear on multiple web pages, or (ii) require information such as seed examples (Etzioni et al., 2005; Wang and Cohen, 2009; Dalvi et al., 2012). These approaches work well for common categories but encounter data sparsity problems for more specific categories, such as the products of a small company or the dishes at a local restaurant. In this context, we may have only a single web page that contains the information we need and no seed examples.

In this paper, we propose a novel task, *zero-shot entity extraction*, where the specification of the desired entities is provided as a natural language query. Given a query (e.g., *hiking trails near Baltimore*) and a web page (e.g., `http://www.everytrail.com/best/hiking-baltimore-maryland`), the goal is to extract all entities corresponding to the query on that page (e.g., *Avalon Super Loop*, etc.). Figure 1 summarizes the task setup.

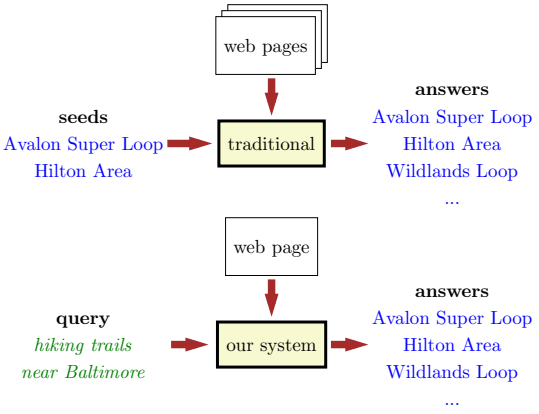


Figure 1: Entity extraction typically requires additional knowledge such as a small set of seed examples or depends on multiple web pages. In our setting, we take as input a natural language query and extract entities from a single web page.

The task introduces two challenges. Given a single web page to extract entities from, we can no longer rely on the redundancy of entities across multiple web pages. Furthermore, in the zero-shot learning paradigm (Larochelle et al., 2008), where entire categories might be unseen during training, the system must generalize to new queries and web pages without the additional aid of seed examples.

To tackle these challenges, we cast the task as a structured prediction problem where the input is the query and the web page, and the output is a list of entities, mediated by a latent *extraction predicate*. To generalize across different inputs, we rely on two types of features: *structural* features, which look at the layout and placement of the entities being extracted; and *denotation* fea-

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 391–401,
Baltimore, Maryland, USA, June 23-25 2014.

tures, which look at the list of entities as a whole and assess their linguistic coherence. When defining features on lists, one technical challenge is being robust to widely varying list sizes. We approach this challenge by defining features over a histogram of abstract tokens derived from the list elements.

For evaluation, we created the OPENWEB dataset comprising natural language queries from the Google Suggest API and diverse web pages returned from web search. Despite the variety of queries and web pages, our system still achieves a test accuracy of 40.5% and an accuracy at 5 of 55.8%.

2 Problem statement

We define the *zero-shot entity extraction* task as follows: let x be a natural language query (e.g., *hiking trails near Baltimore*), and w be a web page. Our goal is to construct a mapping from (x, w) to a list of entities y (e.g., [*Avalon Super Loop*, *Patapsco Valley State Park*, ...]) which are extracted from the web page.

Ideally, we would want our data to be annotated with the correct entity lists y, but this would be very expensive to obtain. We instead define each training and test example as a triple (x, w, c), where the *compatibility function* c maps each y to $c(y) \in \{0, 1\}$ denoting the (approximate) correctness of the list y. In this paper, an entity list y is *compatible* ($c(y) = 1$) when the first, second, and last elements of y match the annotation; otherwise, it is *incompatible* ($c(y) = 0$).

2.1 Dataset

To experiment with a diverse set of queries and web pages, we created a new dataset, OPENWEB, using web pages from Google search results.[1] We use the method from Berant et al. (2013) to generate search queries by performing a breadth-first search over the query space. Specifically, we use the Google Suggest API, which takes a partial query (e.g., *"list of ____ movies"*) and outputs several complete queries (e.g., *"list of horror movies"*). We start with seed partial queries *"list of •____"* where • is one or two initial letters. In each step, we call the Google Suggest API on the partial queries to obtain complete queries,

[1]The OPENWEB dataset and our code base are available for download at `http://www-nlp.stanford.edu/software/web-entity-extractor-ACL2014`.

Full query	New partial queries
list of X IN Y where IN is a preposition (*list of* $[hotels]_X$ *in* $[Guam]_Y$)	*list of* X ____ *list of* ____ X *list of* X IN ____ *list of* ____ IN Y
list of X CC Y where CC is a conjunction (*list of* $[food]_X$ *and* $[drink]_Y$)	*list of* X ____ *list of* ____ X *list of* Y ____ *list of* ____ Y
list of X w (*list of* $[good\ 2012]_X$ $[movies]_w$)	*list of* w ____ *list of* ____ w *list of* X ____

Table 1: Rules for generating new partial queries from complete queries. (X and Y are sequences of words; w is a single word.)

and then apply the transformation rules in Table 1 to generate more partial queries from complete queries. We run the procedure until we obtained 100K queries.

Afterwards, we downloaded the top 2–3 Google search results of each query, sanitized the web pages, and randomly submitted 8000 query / web page pairs to Amazon Mechanical Turk (AMT). Each AMT worker must either mark the web page as irrelevant or extract the first, second, and last entities from the page. We only included examples where at least two AMT workers agreed on the answer.

The resulting OPENWEB dataset consists of 2773 examples from 2269 distinct queries. Among these queries, there are 894 headwords ranging from common categories (e.g., movies, companies, characters) to more specific ones (e.g., enzymes, proverbs, headgears). The dataset contains web pages from 1438 web domains, of which 83% appear only once in our dataset.

Figure 2 shows some queries and web pages from the OPENWEB dataset. Besides the wide range of queries, another main challenge of the dataset comes from the diverse data representation formats, including complex tables, grids, lists, headings, and paragraphs.

3 Approach

Figure 3 shows the framework of our system. Given a query x and a web page w, the system generates a set $\mathcal{Z}(w)$ of *extraction predicates* z which can extract entities from semi-structured data in w. Section 3.1 describes extraction predicates in more detail. Afterwards, the system chooses $z \in \mathcal{Z}(w)$ that maximizes the model probability $p_\theta(z \mid x, w)$, and then executes z on

Queries

airlines of italy
natural causes of global warming
lsu football coaches
bf3 submachine guns
badminton tournaments
foods high in dha
technical colleges in south carolina
songs on glee season 5
singers who use auto tune
san francisco radio stations
actors from boston

Examples (web page, query)

airlines of italy *natural causes of global warming* *lsu football coaches*

Figure 2: Some examples illustrating the diversity of queries and web pages from the OPENWEB dataset.

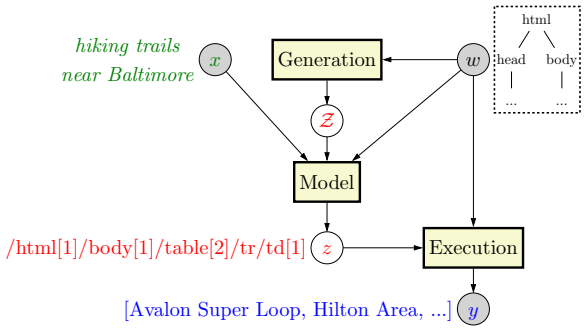


Figure 3: An overview of our system. The system uses the input query x and web page w to produce a list of entities y via an extraction predicate z.

w to get the list of entities $y = [\![z]\!]_w$. Section 3.2 describes the model and the training procedure, while Section 3.3 presents the features used in our model.

3.1 Extraction predicates

We represent each web page w as a DOM tree, a common representation among wrapper induction and web information extraction systems (Sahuguet and Azavant, 1999; Liu et al., 2000; Crescenzi et al., 2001). The text of any DOM tree node that is shorter than 140 characters is a candidate entity. However, without further restrictions, the number of possible entity lists grows exponentially with the number of candidate entities.

To make the problem tractable, we introduce an *extraction predicate* z as an intermediate representation for extracting entities from w. In our system, we let an extraction predicate be a simplified XML path (XPath) such as

```
/html[1]/body[1]/table[2]/tr/td[1]
```

Informally, an extraction predicate is a list of *path entries*. Each path entry is either a tag (e.g., `tr`), which selects all children with that tag; or a tag and an index i (e.g., `td[1]`), which selects only the ith child with that tag. The denotation $y = [\![z]\!]_w$ of an extraction predicate z is the list of entities selected by the XPath. Figure 4 illustrates the execution of the extraction predicate above on a DOM tree.

In the literature, many information extraction systems employ more versatile extraction predicates (Wang and Cohen, 2009; Fumarola et al., 2011). However, despite the simplicity, we are able to find an extraction predicate that extracts a compatible entity list in 69.7% of the development examples. In some examples, we cannot extract a compatible list due to unrecoverable issues such as incorrect annotation. Section 4.4 provides a detailed analysis of these issues. Additionally, extraction predicates can be easily extended to increase the coverage. For example, by introducing new index types `[1:]` (selects all but the first node) and `[:-1]` (selects all but the last node), we can increase the coverage to 76.2%.

Extraction predicate generation. We generate a set $\mathcal{Z}(w)$ of extraction predicates for a given web page w as follows. For each node in the DOM tree, we find an extraction predicate which selects only that node, and then generalizes the predicate by removing any subset of the indices of the last k path entries. For instance, when $k = 2$, an extraction predicate ending in `.../tr[5]/td[2]` will be generalized to `.../tr[5]/td[2]`, `.../tr/td[2]`, `.../tr[5]/td`, and `.../tr/td`. In all experiments, we use $k = 8$, which gives at most 2^8 generalized predicates for each original predicate. This generalization step allows the system to select multiple nodes with the same structure (e.g.,

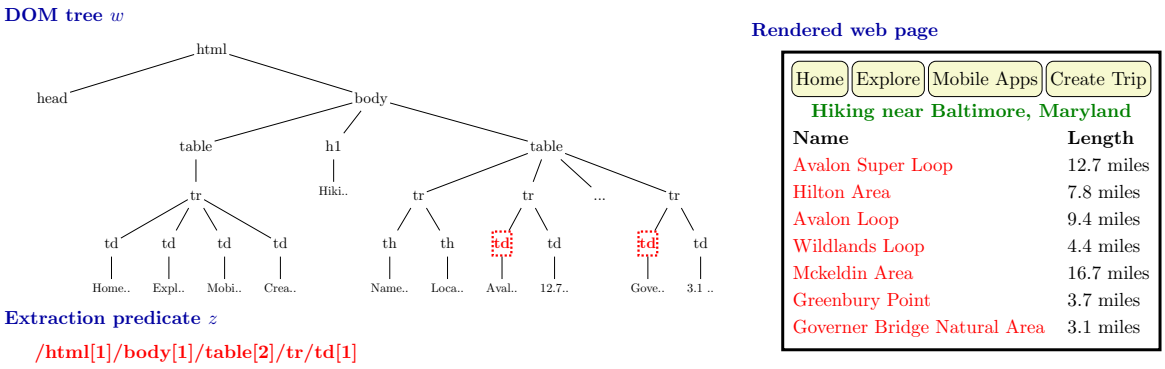


Figure 4: A simplified example of a DOM tree w and an extraction predicate z, which selects a list of entity strings $y = [\![z]\!]_w$ from the page (highlighted in red).

table cells from the same column or list items from the same section of the page).

Out of all generalized extraction predicates, we retain the ones that extract at least two entities from w. Note that several extraction predicates may select the same list of nodes and thus produce the same list of entities.

The procedure above gives a manageable number of extraction predicates. Among the development examples of the OPENWEB dataset, we generate an average of 8449 extraction predicates per example, which evaluate to an average of 1209 unique entity lists.

3.2 Modeling

Given a query x and a web page w, we define a log-linear distribution over all extraction predicates $z \in \mathcal{Z}(w)$ as

$$p_\theta(z \mid x, w) \propto \exp\{\theta^\top \phi(x, w, z)\}, \qquad (1)$$

where $\theta \in \mathbb{R}^d$ is the parameter vector and $\phi(x, w, z)$ is the feature vector, which will be defined in Section 3.3.

To train the model, we find a parameter vector θ that maximizes the regularized log marginal probability of the compatibility function being satisfied. In other words, given training data $\mathcal{D} = \{(x^{(i)}, w^{(i)}, c^{(i)})\}_{i=1}^n$, we find θ that maximizes

$$\sum_{i=1}^{n} \log p_\theta(c^{(i)} = 1 \mid x^{(i)}, w^{(i)}) - \frac{\lambda}{2}\|\theta\|_2^2$$

where

$$p_\theta(c = 1 \mid x, w) = \sum_{z \in \mathcal{Z}(w)} p_\theta(z \mid x, w) \cdot c([\![z]\!]_w).$$

Note that $c([\![z]\!]_w) = 1$ when the entity list $y = [\![z]\!]_w$ selected by z is compatible with the annotation; otherwise, $c([\![z]\!]_w) = 0$.

We use AdaGrad, an online gradient descent with an adaptive per-feature step size (Duchi et al., 2010), making 5 passes over the training data. We use $\lambda = 0.01$ obtained from cross-validation for all experiments.

3.3 Features

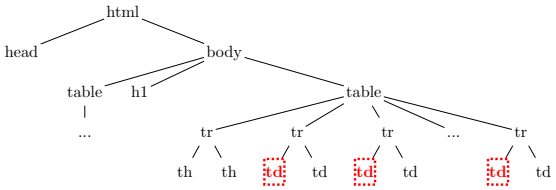
To construct the log-linear model, we define a feature vector $\phi(x, w, z)$ for each query x, web page w, and extraction predicate z. The final feature vector is the concatenation of *structural* features $\phi_{\text{s}}(w, z)$, which consider the selected nodes in the DOM tree, and *denotation* features $\phi_{\text{d}}(x, y)$, which look at the extracted entities.

We will use the query *hiking trails near Baltimore* and the web page in Figure 4 as a running example. Figure 5 lists some features extracted from the example.

3.3.1 Recipe for defining features on lists

One main focus of our work is finding good feature representations for a list of objects (DOM tree nodes for structural features and entity strings for denotation features). One approach is to define the feature vector of a list to be the sum of the feature vectors of individual elements. This is commonly done in structured prediction, where the elements are local configurations (e.g., rule applications in parsing). However, this approach raises a normalization issue when we have to compare and rank lists of drastically different sizes.

As an alternative, we propose a recipe for generating features from a list as follows:



Structural feature	Value
Features on selected nodes:	
TAG-MAJORITY = td	1
INDEX-ENTROPY	0.0
Features on parent nodes:	
CHILDRENCOUNT-MAJORITY = 2	1
PARENT-SINGLE	1
INDEX-ENTROPY	1.0
HEADHOLE (The first node is skipped)	1
Features on grandparent nodes:	
PAGECOVERAGE	0.6
...	...

Selected entities

Avalon Super Loop
Hilton Area
Avalon Loop
Wildlands Loop
Mckeldin Area
Greenbury Point
Governer Bridge Natural Area

Denotation feature	Value
WORDSCOUNT-MEAN	2.42
PHRASESHAPE-MAJORITY = Aa Aa	1
PHRASESHAPE-MAJORITYRATIO	0.71
WORDSHAPE-MAJORITY = Aa	1
PHRASEPOS-MAJORITY = NNP NN	1
LASTWORD-ENTROPY	0.74
WORDPOS = NN (normalized count)	0.53
...	...

Figure 5: A small subset of features from the example *hiking trails near Baltimore* in Figure 4.

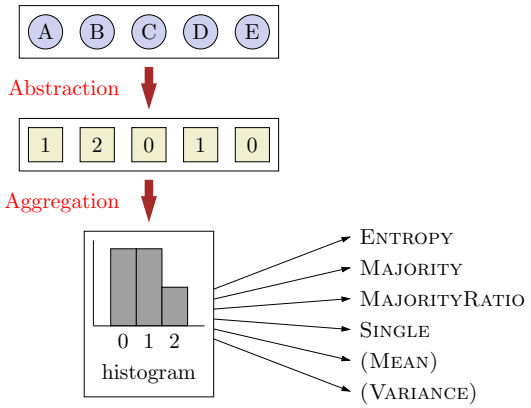


Figure 6: The recipe for defining features on a list of objects: (i) the *abstraction* step converts list elements into abstract tokens; (ii) the *aggregation* step defines features using the histogram of the abstract tokens.

Step 1: Abstraction. We map each list element into an abstract token. For example, we can map each DOM tree node onto an integer equal to the number of children, or map each entity string onto its part-of-speech tag sequence.

Step 2: Aggregation. We create a histogram of the abstract tokens and define features on properties of the histogram. Generally, we use ENTROPY (entropy normalized to the maximum value of 1), MAJORITY (mode), MAJORITYRATIO (percentage of tokens sharing the majority value), and SINGLE (whether all tokens are identical). For abstract tokens with finitely many possible values (e.g., part-of-speech), we also use the normalized histogram count of each possible value as a feature. And for real-valued abstract tokens, we also use the mean and the standard deviation. In the actual system, we convert real-valued features (entropy, histogram count, mean, and standard deviation) into indicator features by binning.

Figure 6 summarizes the steps explained above. We use this recipe for defining both structural and denotation features, which are discussed below.

3.3.2 Structural features

Although different web pages represent data in different formats, they still share some common hierarchical structures in the DOM tree. To capture this, we define structural features $\phi_s(w, z)$, which consider the properties of the selected nodes in the DOM tree, as follows:

Features on selected nodes. We apply our recipe on the list of nodes in w selected by z using the following abstract tokens:

- TAG, ID, CLASS, etc. (HTML attributes)
- CHILDRENCOUNT and SIBLINGSCOUNT (number of children and siblings)
- INDEX (position among its siblings)
- PARENT (parent node; e.g., PARENT-SINGLE means that all nodes share the same parent.)

Additionally, we define the following features based on the coverage of all selected nodes:

- NOHOLE, HEADHOLE, etc. (node coverage in the same DOM tree level; e.g., HEADHOLE activates when the first sibling of the selected nodes is not selected.)

- PAGECOVERAGE (node coverage relative to the entire tree; we use depth-first traversal timestamps to estimate the fraction of nodes in the subtrees of the selected nodes.)

Features on ancestor nodes. We also define the same feature set on the list of ancestors of the selected nodes in the DOM tree. In our experiments, we traverse up to 5 levels of ancestors and define features from the nodes in each level.

3.3.3 Denotation features

Structural features are not powerful enough to distinguish between entity lists appearing in similar structures such as columns of the same table or fields of the same record. To solve this ambiguity, we introduce denotation features $\phi_{\mathrm{d}}(x, y)$ which considers the coherence or appropriateness of the selected entity strings $y = [\![z]\!]_w$.

We observe that the correct entities often share some linguistic statistics. For instance, entities in many categories (e.g., people and place names) usually have only 2–3 word tokens, most of which are proper nouns. On the other hand, random words on the web page tend to have more diverse lengths and part-of-speech tags.

We apply our recipe on the list of selected entities using the following abstract tokens:

- WORDSCOUNT (number of words)
- PHRASESHAPE (abstract shape of the phrase; e.g., *Barack Obama* becomes `Aa Aa`)
- WORDSHAPE (abstract shape of each word; the number of abstract tokens will be the total number of words over all selected entities)
- FIRSTWORD and LASTWORD
- PHRASEPOS and WORDPOS (part-of-speech tags for whole phrases and individual words)

4 Experiments

In this section we evaluate our system on the OPENWEB dataset.

4.1 Evaluation metrics

Accuracy. As the main metric, we use a notion of *accuracy* based on compatibility; specifically, we define the accuracy as the fraction of examples where the system predicts a compatible entity list as defined in Section 2. We also report *accuracy at 5*, the fraction of examples where the top five predictions contain a compatible entity list.

Path suffix pattern (multiset)	Count
{a, table, tbody, td[*], tr}	1792
{a, tbody, td[*], text, tr}	1591
{a, table[*], tbody, td[*], tr}	1325
{div, table, tbody, td[*], tr}	1259
{b, div, div, div, div[*]}	1156
{div[*], table, tbody, td[*], tr}	1059
{div, table[*], tbody, td[*], tr}	844
{table, tbody, td[*], text, tr}	828
{div[*], table[*], tbody, td[*], tr}	793
{a, table, tbody, td, tr}	743

Table 2: Top 10 path suffix patterns found by the baseline learner in the development data. Since we allow path entries to be permuted, each suffix pattern is represented by a multiset of path entries. The notation `[*]` denotes any path entry index.

To see how our compatibility-based accuracy tracks exact correctness, we sampled 100 web pages which have at least one valid extraction predicate and manually annotated the full list of entities. We found that in 85% of the examples, the longest compatible list y is the correct list of entities, and many lists in the remaining 15% miss the correct list by only a few entities.

Oracle. In some examples, our system cannot find any list of entities that is compatible with the gold annotation. The *oracle* score is the fraction of examples in which the system can find at least one compatible list.

4.2 Baseline

As a baseline, we list the suffixes of the correct extraction predicates in the training data, and then sort the resulting suffix patterns by frequency. To improve generalization, we treat path entries with different indices (e.g., `td[1]` vs. `td[2]`) as equivalent and allow path entries to be permuted. Table 2 lists the top 10 suffix patterns from the development data. At test time, we choose an extraction predicate with the most frequent suffix pattern. The baseline should work considerably well if the web pages were relatively homogeneous.

4.3 Main results

We held out 30% of the dataset as test data. For the results on development data, we report the average across 10 random 80-20 splits. Table 3 shows the results. The system gets an accuracy of 41.1% and 40.5% for the development and test data, respectively. If we consider the top 5 lists of entities, the accuracy increases to 58.4% on the development data and 55.8% on the test data.

	Development data		Test data	
	Acc	**A@5**	**Acc**	**A@5**
Baseline	10.8 ± 1.3	25.6 ± 2.0	10.3	20.9
Our system	41.1 ± 3.4	58.4 ± 2.7	40.5	55.8
Oracle	68.7 ± 2.4	68.7 ± 2.4	66.6	66.6

Table 3: Main results on the OPENWEB dataset using the default set of features. (Acc = accuracy, A@5 = accuracy at 5)

4.4 Error analysis

We now investigate the errors made by our system using the development data. We classify the errors into two types: (i) *coverage errors*, which are when the system cannot find any entity list satisfying the compatibility function; and (ii) *ranking errors*, which are when a compatible list of entities exists, but the system outputs an incompatible list.

Tables 4 and 5 show the breakdown of coverage and ranking errors from an experiment on the development data.

Analysis of coverage errors. From Table 4, about 36% of coverage errors happen when the extraction predicate for the correct entities also captures unrelated parts of the web page (Reason C1). For example, many Wikipedia articles have the *See Also* section that lists related articles in an unordered list (`/ul/li/a`), which causes a problem when the entities are also represented in the same format.

Another main source of errors is the inconsistency in HTML tag usage (Reason C2). For instance, some web pages use `<b>` and `<strong>` tags for bold texts interchangeably, or switch between `<b><a>...</a></b>` and `<a><b>...</b></a>` across entities. We expect that this problem can be solved by normalizing the web page, using an alternative web page representation (Cohen et al., 2002; Wang and Cohen, 2009; Fumarola et al., 2011), or leveraging more expressive extraction predicates (Dalvi et al., 2011).

One interesting source of errors is Reason C3, where we need to *filter* the selected entities to match the complex requirement in the query. For example, the query *tech companies in China* requires the system to select only the company names with *China* in the corresponding location column. To handle such queries, we need a deeper understanding of the relation between the linguistic structure of the query and the hierarchical structure of the web page. Tackling this error requires compositionality and is critical to generalize to more complex queries.

Setting	**Acc**	**A@5**
All features	41.1 ± 3.4	58.4 ± 2.7
Oracle	68.7 ± 2.4	68.7 ± 2.4
(Section 4.5)		
Structural features only	36.2 ± 1.9	54.5 ± 2.5
Denotation features only	19.8 ± 2.5	41.7 ± 2.7
(Section 4.6)		
Structural + query-denotation	41.7 ± 2.5	58.1 ± 2.4
Query-denotation features only	25.0 ± 2.3	48.0 ± 2.7
Concat. a random web page + structural + denotation	19.3 ± 2.6	41.2 ± 2.3
Concat. a random web page + structural + query-denotation	29.2 ± 1.7	49.2 ± 2.2
(Section 4.7)		
Add 1 seed entity	52.9 ± 3.0	66.5 ± 2.5

Table 6: System accuracy with different feature and input settings on the development data. (Acc = accuracy, A@5 = accuracy at 5)

Analysis of ranking errors. From Table 5, a large number of errors are attributed to the system selecting non-content elements such as navigation links and content headings (Reason R1). Feature analysis reveals that both structural and linguistic statistics of these non-content elements can be more coherent than those of the correct entities. We suspect that since many of our features try to capture the coherence of entities, the system sometimes erroneously favors the more homonogenous non-content parts of the page. To disfavor these parts, One possible solution is to add visual features that capture how the web page is rendered and favor more salient parts of the page. (Liu et al., 2003; Song et al., 2004; Zhu et al., 2005; Zheng et al., 2007).

4.5 Feature variations

We now investigate the contribution of each feature type. The ablation results on the development set over 10 random splits are shown in Table 6. We observe that denotation features improves accuracy on top of structural features.

Table 7 shows an example of an error that is eliminated by each feature type. Generally, if the entities are represented as records (e.g., rows of a table), then denotation features will help the system select the correct field from each record. On the other hand, structural features prevent the system from selecting random entities outside the main part of the page.

Reason		Short example	Count
C1	Answers and contextual elements are selected by the same extraction predicate.	Select entries in *See Also* section in addition to the content because they are all list entries.	48
C2	HTML tag usage is inconsistent.	The page uses both `b` and `strong` for headers.	16
C3	The query applies to only some sections of the matching entities.	Need to select only companies in China from the table of all Asian companies.	20
C4	Answers are embedded in running text.	Answers are in a comma-separated list.	13
C5	Text normalization issues.	Selected *Silent Night Lyrics* instead of *Silent Night*.	19
C6	Other issues.	Incorrect annotation. / Entities are permuted when the web page is rendered. / etc.	18
Total			134

Table 4: Breakdown of coverage errors from the development data.

Reason		Short example	Count
R1	Select non-content strings.	Select navigation links, headers, footers, or sidebars.	25
R2	Select entities from a wrong field.	Select book authors instead of book names.	22
R3	Select entities from the wrong section(s).	For the query *schools in Texas*, select all schools on the page, or select the schools in Alabama instead.	19
R4	Also select headers or footers.	Select the table header in addition to the answers.	7
R5	Select only entities with a particular formatting.	From a list of answers, select only anchored (`a`) entities.	4
R6	Select headings instead of the contents or vice versa.	Select the categories of rums in `h2` tags instead of the rum names in the tables.	2
R7	Other issues.	Incorrect annotation. / Multiple sets of answers appear on the same page. / etc.	9
Total			88

Table 5: Breakdown of ranking errors from the development data.

All features	Structural only	Denotation only
The Sun	CIRC: 2,279,492	Paperboy Australia
Daily Mail	CIRC: 1,821,684	Paperboy UK
Daily Mirror	CIRC: 1,032,144	Paperboy Home Page
...	...	...

Table 7: System outputs for the query *UK newspapers* with different feature sets. Without denotation features, the system selects the daily circulation of each newspaper instead of the newspaper names. And without structural features, the system selects the hidden navigation links from the top of the page.

4.6 Incorporating query information

So far, note that all our features depend only on the extraction predicate z and not the input query x. Remarkably, we were still able to obtain reasonable results. One explanation is that since we obtained the web pages from a search engine, the most prominent entities on the web pages, such as entities in table cells in the middle of the page, are likely to be good independent of the query.

However, different queries often denote entities with different linguistic properties. For example, queries *mayors of Chicago* and *universities in Chicago* will produce entities of different lengths, part-of-speech sequences, and word distributions. This suggests incorporating features that depend on the query.

To explore the potential of query information, we conduct the following oracle experiment. We replace each denotation feature $f(y)$ with a corresponding query-denotation feature $(f(y), g(x))$, where $g(x)$ is the category of the query x. We manually classified all queries in our dataset into 7 categories: person, media title, location/organization, abtract entity, word/phrase, object name, and miscellaneous.

Table 8 shows some examples where adding these query-denotation features improves the selected entity lists by favoring answers that are more suitable to the query category. However, Table 6 shows that these new features do not significantly improve the accuracy of our original system on the development data.

We suspect that any gains offered by the query-denotation features are subsumed by the structural features. To test this hypothesis, we conducted two experiments, the results of which are shown in Table 6. First, we removed structural features and found that using query-denotation features improves accuracy significantly over using denotation features alone from 19.8% to 25.0%. Second, we created a modified dataset where the web page in each example is a concatenation of the original web page and an unrelated web page. On

Query	euclid's elements book titles	soft drugs	professional athletes with concussions
Default features	"Prematter", "Book I.", "Book II.", "Book III.", ...	"Hard drugs", "Soft drugs", "Some drugs cannot be classified that way", ...	"Pistons-Knicks Game Becomes Site of Incredible Dance Battle", "Toronto Mayor Rob Ford Attends ...", ...
Structural + Query-Denotation	(category = *media title*) "Book I. The fundamentals ...", "Book II. Geometric algebra", ...	(category = *object name*) "methamphetamine", "psilocybin", "caffeine"	(category = *person*) "Mike Richter", "Stu Grimson", "Geoff Courtnall", ...

Table 8: System outputs after changing denotation features into query-denotation features.

this modified dataset, the prominent entities may not be the answers to the query. Here, query-denotation features improves accuracy over denotation features alone from 19.3% to 29.2%.

4.7 Comparison with other problem settings

Since zero-shot entity extraction is a new task, we cannot directly compare our system with other systems. However, we can mimic the settings of other tasks. In one experiment, we augment each input query with a single seed entity (the second annotated entity in our experiments); this setting is suggestive of Wang and Cohen (2009). Table 6 shows that this augmentation increases accuracy from 41.1% to 52.9%, suggesting that our system can perform substantially better with a small amount of additional supervision.

5 Discussion

Our work shares a base with the wrapper induction literature (Kushmerick, 1997) in that it leverages regularities of web page structures. However, wrapper induction usually focuses on a small set of web domains, where the web pages in each domain follow a fixed template (Muslea et al., 2001; Crescenzi et al., 2001; Cohen et al., 2002; Arasu and Garcia-Molina, 2003). Later work in web data extraction attempts to generalize across different web pages, but relies on either restricted data formats (Wong et al., 2009) or prior knowledge of web page structures with respect to the type of data to extract (Zhang et al., 2013).

In our case, we only have the natural language query, which presents the more difficult problem of associating the entity class in the query (e.g., *hiking trails*) to concrete entities (e.g., *Avalon Super Loop*). In contrast to information extraction systems that extract homogeneous records from web pages (Liu et al., 2003; Zheng et al., 2009), our system must choose the correct field from each record and also identify the relevant part of the page based on the query.

Another related line of work is information extraction from text, which relies on natural language patterns to extract categories and relations of entities. One classic example is Hearst patterns (Hearst, 1992; Etzioni et al., 2005), which can learn new entities and extraction patterns from seed examples. More recent approaches also leverage semi-structured data to obtain more robust extraction patterns (Mintz et al., 2009; Hoffmann et al., 2011; Surdeanu et al., 2012; Riedel et al., 2013). Although our work focuses on semi-structured web pages rather than raw text, we use linguistic patterns of queries and entities as a signal for extracting appropriate answers.

Additionally, our efforts can be viewed as building a lexicon on the fly. In recent years, there has been a drive to scale semantic parsing to large databases such as Freebase (Cai and Yates, 2013; Berant et al., 2013; Kwiatkowski et al., 2013). However, despite the best efforts of information extraction, such databases will always lag behind the open web. For example, Berant et al. (2013) found that less than 10% of naturally occurring questions are answerable by a simple Freebase query. By using the semi-structured data from the web as a knowledge base, we hope to increase fact coverage for semantic parsing.

Finally, as pointed out in the error analysis, we need to filter or aggregate the selected entities for complex queries (e.g., *tech companies in China* for a web page with all Asian tech companies). In future work, we would like to explore the issue of compositionality in queries by aligning linguistic structures in natural language with the relative position of entities on web pages.

Acknowledgements

We gratefully acknowledge the support of the Google Natural Language Understanding Focused Program. In addition, we would like to thank anonymous reviewers for their helpful comments.

References

A. Arasu and H. Garcia-Molina. 2003. Extracting structured data from web pages. In *ACM SIGMOD international conference on Management of data*, pages 337–348.

J. Berant, A. Chou, R. Frostig, and P. Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Q. Cai and A. Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *Association for Computational Linguistics (ACL)*.

W. W. Cohen, M. Hurst, and L. S. Jensen. 2002. A flexible learning system for wrapping tables and lists in HTML documents. In *World Wide Web (WWW)*, pages 232–241.

V. Crescenzi, G. Mecca, P. Merialdo, et al. 2001. Roadrunner: Towards automatic data extraction from large web sites. In *VLDB*, volume 1, pages 109–118.

N. Dalvi, R. Kumar, and M. Soliman. 2011. Automatic wrappers for large scale web extraction. *Proceedings of the VLDB Endowment*, 4(4):219–230.

B. Dalvi, W. Cohen, and J. Callan. 2012. Websets: Extracting sets of entities from the web using unsupervised information extraction. In *Web Search and Data Mining (WSDM)*, pages 243–252.

J. Duchi, E. Hazan, and Y. Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. In *Conference on Learning Theory (COLT)*.

O. Etzioni, M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134.

F. Fumarola, T. Weninger, R. Barber, D. Malerba, and J. Han. 2011. *Extracting general lists from web documents: A hybrid approach*. Modern Approaches in Applied Intelligence Springer.

M. A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Interational Conference on Computational linguistics*, pages 539–545.

R. Hoffmann, C. Zhang, X. Ling, L. S. Zettlemoyer, and D. S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Association for Computational Linguistics (ACL)*, pages 541–550.

N. Kushmerick. 1997. *Wrapper induction for information extraction*. Ph.D. thesis, University of Washington.

T. Kwiatkowski, E. Choi, Y. Artzi, and L. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Empirical Methods in Natural Language Processing (EMNLP)*.

H. Larochelle, D. Erhan, and Y. Bengio. 2008. Zero-data learning of new tasks. In *AAAI*, volume 8, pages 646–651.

L. Liu, C. Pu, and W. Han. 2000. XWRAP: An XML-enabled wrapper construction system for web information sources. In *Data Engineering, 2000. Proceedings. 16th International Conference on*, pages 611–621.

B. Liu, R. Grossman, and Y. Zhai. 2003. Mining data records in web pages. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–606.

M. Mintz, S. Bills, R. Snow, and D. Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Association for Computational Linguistics (ACL)*, pages 1003–1011.

I. Muslea, S. Minton, and C. A. Knoblock. 2001. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems*, 4(1):93–114.

S. Riedel, L. Yao, and A. McCallum. 2013. Relation extraction with matrix factorization and universal schemas. In *North American Association for Computational Linguistics (NAACL)*.

A. Sahuguet and F. Azavant. 1999. WysiWyg web wrapper factory (W4F). In *WWW Conference*.

R. Song, H. Liu, J. Wen, and W. Ma. 2004. Learning block importance models for web pages. In *World Wide Web (WWW)*, pages 203–211.

M. Surdeanu, J. Tibshirani, R. Nallapati, and C. D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*, pages 455–465.

R. C. Wang and W. W. Cohen. 2009. Character-level analysis of semi-structured documents for set expansion. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1503–1512.

Y. W. Wong, D. Widdows, T. Lokovic, and K. Nigam. 2009. Scalable attribute-value extraction from semi-structured text. In *IEEE International Conference on Data Mining Workshops*, pages 302–307.

Z. Zhang, K. Q. Zhu, H. Wang, and H. Li. 2013. Automatic extraction of top-k lists from the web. In *International Conference on Data Engineering*.

S. Zheng, R. Song, and J. Wen. 2007. Template-independent news extraction based on visual consistency. In *AAAI*, volume 7, pages 1507–1513.

S. Zheng, R. Song, J. Wen, and C. L. Giles. 2009. Efficient record-level wrapper induction. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 47–56.

J. Zhu, Z. Nie, J. Wen, B. Zhang, and W. Ma. 2005. 2D conditional random fields for web information extraction. In *International Conference on Machine Learning (ICML)*, pages 1044–1051.

Incremental Joint Extraction of Entity Mentions and Relations

Qi Li **Heng Ji**
Computer Science Department
Rensselaer Polytechnic Institute
Troy, NY 12180, USA
{liq7,jih}@rpi.edu

Abstract

We present an incremental joint framework to simultaneously extract entity mentions and relations using structured perceptron with efficient beam-search. A segment-based decoder based on the idea of semi-Markov chain is adopted to the new framework as opposed to traditional token-based tagging. In addition, by virtue of the inexact search, we developed a number of new and effective global features as soft constraints to capture the interdependency among entity mentions and relations. Experiments on Automatic Content Extraction (ACE)[1] corpora demonstrate that our joint model significantly outperforms a strong pipelined baseline, which attains better performance than the best-reported end-to-end system.

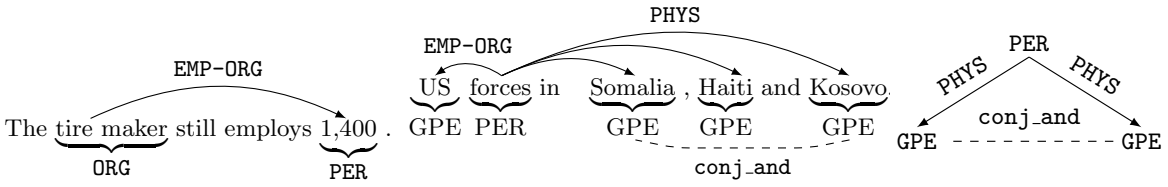
1 Introduction

The goal of end-to-end entity mention and relation extraction is to discover relational structures of entity mentions from unstructured texts. This problem has been artificially broken down into several components such as entity mention boundary identification, entity type classification and relation extraction. Although adopting such a pipelined approach would make a system comparatively easy to assemble, it has some limitations: First, it prohibits the interactions between components. Errors in the upstream components are propagated to the downstream components without any feedback. Second, it over-simplifies the problem as multiple local classification steps without modeling long-distance and cross-task dependencies. By contrast, we re-formulate this task as a structured prediction problem to reveal the linguistic and logical properties of the hidden structures. For example, in Figure 1, the output structure of each sentence can be interpreted as a graph in which entity mentions are nodes and relations are directed arcs with relation types. By jointly predicting the structures, we aim to address the aforementioned limitations by capturing: (i) The interactions between two tasks. For example, in Figure 1a, although it may be difficult for a mention extractor to predict "1,400" as a Person (PER) mention, the context word "employs" between "tire maker" and "1,400" strongly indicates an Employment-Organization (EMP-ORG) relation which must involve a PER mention. (ii) The global features of the hidden structure. Various entity mentions and relations share linguistic and logical constraints. For example, we can use the triangle feature in Figure 1b to ensure that the relations between "forces", and each of the entity mentions "Somalia$_{\text{GPE}}$", "Haiti$_{\text{GPE}}$" and "Kosovo$_{\text{GPE}}$", are of the same type (Physical (PHYS), in this case).

Following the above intuitions, we introduce a joint framework based on structured perceptron (Collins, 2002; Collins and Roark, 2004) with beam-search to extract entity mentions and relations simultaneously. With the benefit of inexact search, we are also able to use arbitrary global features with low cost. The underlying learning algorithm has been successfully applied to some other Natural Language Processing (NLP) tasks. Our task differs from dependency parsing (such as (Huang and Sagae, 2010)) in that relation structures are more flexible, where each node can have arbitrary relation arcs. Our previous work (Li et al., 2013) used perceptron model with token-based tagging to jointly extract event triggers and arguments. By contrast, we aim to address a more challenging task: identifying mention boundaries and types together with relations, which raises the issue that assignments for the same sentence with different mention boundaries are difficult to syn-

[1] http://www.itl.nist.gov/iad/mig//tests/ace

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 402–412,
Baltimore, Maryland, USA, June 23-25 2014.



(a) Interactions between Two Tasks (b) Example of Global Feature

Figure 1: End-to-End Entity Mention and Relation Extraction.

chronize during search. To tackle this problem, we adopt a segment-based decoding algorithm derived from (Sarawagi and Cohen, 2004; Zhang and Clark, 2008) based on the idea of semi-Markov chain (a.k.a, multiple-beam search algorithm).

Most previous attempts on joint inference of entity mentions and relations (such as (Roth and Yih, 2004; Roth and Yih, 2007)) assumed that entity mention boundaries were given, and the classifiers of mentions and relations are separately learned. As a key difference, we incrementally extract entity mentions together with relations using a single model. The main contributions of this paper are as follows:

1. This is the first work to incrementally predict entity mentions and relations using a single joint model (Section 3).
2. Predicting mention boundaries in the joint framework raises the challenge of synchronizing different assignments in the same beam. We solve this problem by detecting entity mentions on segment-level instead of traditional token-based approaches (Section 3.1.1).
3. We design a set of novel global features based on soft constraints over the entire output graph structure with low cost (Section 4).

Experimental results show that the proposed framework achieves better performance than pipelined approaches, and global features provide further significant gains.

2 Background

2.1 Task Definition

The entity mention extraction and relation extraction tasks we are addressing are those of the Automatic Content Extraction (ACE) program[2]. ACE defined 7 main entity types including Person (PER), Organization (ORG), Geographical Entities (GPE), Location (LOC), Facility (FAC), Weapon (WEA) and Vehicle (VEH). The goal of relation extraction[3] is to extract semantic relations of the targeted types between a pair of entity mentions which appear in the same sentence. ACE'04 defined 7 main relation types: Physical (PHYS), Person-Social (PER-SOC), Employment-Organization (EMP-ORG), Agent-Artifact (ART), PER/ORG Affiliation (Other-AFF), GPE-Affiliation (GPE-AFF) and Discourse (DISC). ACE'05 kept PER-SOC, ART and GPE-AFF, split PHYS into PHYS and a new relation type Part-Whole, removed DISC, and merged EMP-ORG and Other-AFF into EMP-ORG.

Throughout this paper, we use $\perp$ to denote non-entity or non-relation classes. We consider relation asymmetric. The same relation type with opposite directions is considered to be two classes, which we refer to as *directed relation types*.

Most previous research on relation extraction assumed that entity mentions were given In this work we aim to address the problem of end-to-end entity mention and relation extraction from raw texts.

2.2 Baseline System

In order to develop a baseline system representing state-of-the-art pipelined approaches, we trained a linear-chain Conditional Random Fields model (Lafferty et al., 2001) for entity mention extraction and a Maximum Entropy model for relation extraction.

Entity Mention Extraction Model We re-cast the problem of entity mention extraction as a sequential token tagging task as in the state-of-the-art system (Florian et al., 2006). We applied the BILOU scheme, where each tag means a token is the **B**eginning, **I**nside, **L**ast, **O**utside, and **U**nit of an entity mention, respectively. Most of our features are similar to the work of (Florian et al.,

[2]http://www.nist.gov/speech/tests/ace

[3]Throughout this paper we refer to relation mention as relation since we do not consider relation mention coreference.

2004; Florian et al., 2006) except that we do not have their gazetteers and outputs from other mention detection systems as features. Our additional features are as follows:

- Governor word of the current token based on dependency parsing (Marneffe et al., 2006).
- Prefix of each word in Brown clusters learned from TDT5 corpus (Sun et al., 2011).

Relation Extraction Model Given a sentence with entity mention annotations, the goal of baseline relation extraction is to classify each mention pair into one of the pre-defined relation types with direction or $\perp$ (non-relation). Most of our relation extraction features are based on the previous work of (Zhou et al., 2005) and (Kambhatla, 2004). We designed the following additional features:

- The label sequence of phrases covering the two mentions. For example, for the sentence in Figure 1a, the sequence is "NP VP NP". We also augment it by head words of each phrase.
- Four syntactico - semantic patterns described in (Chan and Roth, 2010).
- We replicated each lexical feature by replacing each word with its Brown cluster.

3 Algorithm

3.1 The Model

Our goal is to predict the hidden structure of each sentence based on arbitrary features and constraints. Let $x \in \mathcal{X}$ be an input sentence, $y' \in \mathcal{Y}$ be a candidate structure, and $\mathbf{f}(x, y')$ be the feature vector that characterizes the entire structure. We use the following linear model to predict the most probable structure $\hat{y}$ for x:

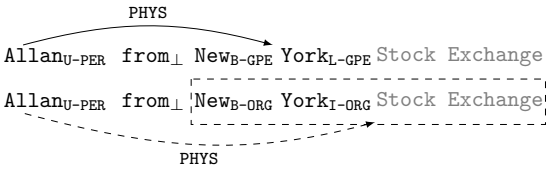
$$\hat{y} = \underset{y' \in \mathcal{Y}(x)}{\operatorname{argmax}} \mathbf{f}(x, y') \cdot \mathbf{w} \quad (1)$$

where the score of each candidate assignment is defined as the inner product of the feature vector $\mathbf{f}(x, y')$ and feature weights $\mathbf{w}$.

Since the structures contain both entity mentions relations, and we also aim to exploit global features. There does not exist a polynomial-time algorithm to find the best structure. In practice we apply beam-search to expand partial configurations for the input sentence incrementally to find the structure with the highest score.

3.1.1 Joint Decoding Algorithm

One main challenge to search for entity mentions and relations incrementally is the alignment of different assignments. Assignments for the same sentence can have different numbers of entity mentions and relation arcs. The entity mention extraction task is often re-cast as a token-level sequential labeling problem with BIO or BILOU scheme (Ratinov and Roth, 2009; Florian et al., 2006). A naive solution to our task is to adopt this strategy by treating each token as a state. However, different assignments for the same sentence can have various mention boundaries. It is unfair to compare the model scores of a partial mention and a complete mention. It is also difficult to synchronize the search process of relations. For example, consider the two hypotheses ending at "York" for the same sentence:



The model would bias towards the incorrect assignment "New/B-GPE York/L-GPE" since it can have more informative features as a complete mention (e.g., a binary feature indicating if the entire mention appears in a GPE gazetter). Furthermore, the predictions of the two PHYS relations cannot be synchronized since "New/B-FAC York/I-FAC" is not yet a complete mention.

To tackle these problems, we employ the idea of semi-Markov chain (Sarawagi and Cohen, 2004), in which each state corresponds to a segment of the input sequence. They presented a variant of Viterbi algorithm for exact inference in semi-Markov chain. We relax the max operation by beam-search, resulting in a segment-based decoder similar to the multiple-beam algorithm in (Zhang and Clark, 2008). Let $\hat{d}$ be the upper bound of entity mention length. The k-*best* partial assignments ending at the i-th token can be calculated as:

$$B[i] = \underset{y' \in \{y_{[1..i]} | y_{[1:i-d]} \in B[i-d],\, d=1...\hat{d}\}}{\text{k-}BEST} \mathbf{f}(x, y') \cdot \mathbf{w}$$

where $y_{[1:i-d]}$ stands for a partial configuration ending at the $(i\text{-}d)$-th token, and $y_{[i-d+1,i]}$ corresponds to the structure of a new segment (i.e., subsequence of x) $x_{[i-d+1,i]}$. Our joint decoding algorithm is shown in Figure 2. For each token index i, it maintains a beam for the partial assignments whose last segments end at the i-th token. There are two types of actions during the search:

Input: input sentence $x = (x_1, x_2, ..., x_m)$.
k: beam size.
$\mathcal{T} \cup \{\perp\}$: entity mention type alphabet.
$\mathcal{R} \cup \{\perp\}$: directed relation type alphabet.[4]
d_t: max length of type-t segment, $t \in \mathcal{T} \cup \{\perp\}$.
Output: best configuration $\hat{y}$ for x

```
1  initialize m empty beams B[1..m]
2  for i ← 1...m do
3    for t ∈ T ∪ {⊥} do
4      for d ← 1...d_t, y' ∈ B[i − d] do
5        k ← i − d + 1
6        B[i] ← B[i] ∪ APPEND(y', t, k, i)
7    B[i] ← k-BEST(B[i])
8    for j ← (i − 1)...1 do
9      buf ← ∅
10     for y' ∈ B[i] do
11       if HASPAIR(y', i, j) then
12         for r ∈ R ∪ {⊥} do
13           buf ← buf ∪ LINK(y', r, i, j)
14       else
15         buf ← buf ∪ {y'}
16     B[i] ← k-BEST(buf)
17 return B[m][0]
```

Figure 2: Joint Decoding for Entity Mentions and Relations. *HASPAIR*(y', i, j) checks if there are two entity mentions in y' that end at token x_i and token x_j, respectively. *APPEND*(y', t, k, i) appends y' with a type-t segment spanning from x_k to x_i. Similarly *LINK*(y', r, i, j) augments y' by assigning a directed relation r to the pair of entity mentions ending at x_i and x_j respectively.

1. *APPEND* (Lines 3-7). First, the algorithm enumerates all possible segments (i.e., subsequences) of x ending at the current token with various entity types. A special type of segment is a single token with non-entity label ($\perp$). Each segment is then appended to existing partial assignments in one of the previous beams to form new assignments. Finally the top k results are recorded in the current beam.
2. *LINK* (Lines 8-16). After each step of *APPEND*, the algorithm looks backward to link the newly identified entity mentions and previous ones (if any) with relation arcs. At the j-th sub-step, it only considers the previous mention ending at the j-th previous token. Therefore different configurations are guaranteed to have the same number of sub-steps. Finally, all assignments are re-ranked with new relation information.

[4]The same relation type with opposite directions is considered to be two classes in $\mathcal{R}$.

There are m *APPEND* actions, each is followed by at most $(i-1)$ *LINK* actions (line 8). Therefore the worst-case time complexity is $O(\hat{d} \cdot k \cdot m^2)$, where $\hat{d}$ is the upper bound of segment length.

3.1.2 Example Demonstration

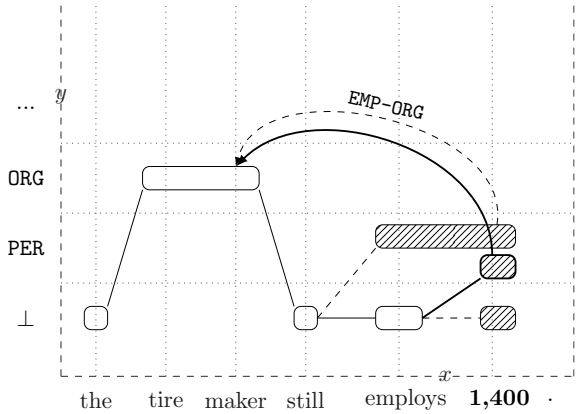


Figure 3: Example of decoding steps. x-axis and y-axis represent the input sentence and entity types, respectively. The rectangles denote segments with entity types, among which the shaded ones are three competing hypotheses ending at "1,400". The solid lines and arrows indicate correct *APPEND* and *LINK* actions respectively, while the dashed indicate incorrect actions.

Here we demonstrate a simple but concrete example by considering again the sentence described in Figure 1a. Suppose we are at the token "1,400". At this point we can propose multiple entity mentions with various lengths. Assuming "1,400$_{/\texttt{PER}}$", "1,400$_{/\perp}$" and "(employs 1,400)$_{/\texttt{PER}}$" are possible assignments, the algorithm appends these new segments to the partial assignments in the beams of the tokens "employs" and "still", respectively. Figure 3 illustrates this process. For simplicity, only a small part of the search space is presented. The algorithm then links the newly identified mentions to the previous ones in the same configuration. In this example, the only previous mention is "(tire maker)$_{/\texttt{ORG}}$". Finally, "1,400$_{/\texttt{PER}}$" will be preferred by the model since there are more indicative context features for EMP−ORG relation between "(tire maker)$_{/\texttt{PER}}$" and "1,400$_{/\texttt{PER}}$".

3.2 Structured-Perceptron Learning

To estimate the feature weights, we use structured perceptron (Collins, 2002), an extension of the standard perceptron for structured prediction, as the learning framework. Huang et al. (2012) proved the convergency of structured perceptron when inexact search is applied with violation-fixing update methods such as early-update (Collins and Roark, 2004). Since we use beam-search in this work, we apply early-update. In addition, we use averaged parameters to reduce overfitting as in (Collins, 2002).

Figure 4 shows the pseudocode for structured perceptron training with early-update. Here *BEAMSEARCH* is identical to the decoding algorithm described in Figure 2 except that if y', the prefix of the gold standard y, falls out of the beam after each execution of the k-*BEST* function (line 7 and 16), then the top assignment z and y' are returned for parameter update. It is worth noting that this can only happen if the gold-standard has a segment ending at the current token. For instance, in the example of Figure 1a, $B[2]$ cannot trigger any early-update since the gold standard does not contain any segment ending at the second token.

Input: training set $\mathcal{D} = \{(x^{(j)}, y^{(j)})\}_{i=1}^{N}$, maximum iteration number T
Output: model parameters $\mathbf{w}$
1 initialize $\mathbf{w} \leftarrow \mathbf{0}$
2 **for** $t \leftarrow 1...T$ **do**
3 **foreach** $(x, y) \in \mathcal{D}$ **do**
4 $(x, y', z) \leftarrow$ *BEAMSEARCH* $(x, y, \mathbf{w})$
5 **if** $z \neq y$ **then**
6 $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{f}(x, y') - \mathbf{f}(x, z)$
7 return $\mathbf{w}$

Figure 4: Perceptron algorithm with beam-search and early-update. y' is the prefix of the gold-standard and z is the top assignment.

3.3 Entity Type Constraints

Entity type constraints have been shown effective in predicting relations (Roth and Yih, 2007; Chan and Roth, 2010). We automatically collect a mapping table of permissible entity types for each relation type from our training data. Instead of applying the constraints in post-processing inference, we prune the branches that violate the type constraints during search. This type of pruning can reduce search space as well as make the input for parameter update less noisy. In our experiments, only 7 relation mentions (0.5%) in the dev set and 5 relation mentions (0.3%) in the test set violate the constraints collected from the training data.

4 Features

An advantage of our framework is that we can easily exploit arbitrary features across the two tasks. This section describes the local features (Section 4.1) and global features (Section 4.2) we developed in this work.

4.1 Local Features

We design segment-based features to directly evaluate the properties of an entity mention instead of the individual tokens it contains. Let $\hat{y}$ be a predicted structure of a sentence x. The entity segments of $\hat{y}$ can be expressed as a list of triples $(e_1, ..., e_m)$, where each segment $e_i = \langle u_i, v_i, t_i \rangle$ is a triple of start index u_i, end index v_i, and entity type t_i. The following is an example of segment-based feature:

$$f_{001}(x, \hat{y}, i) = \begin{cases} 1 & \text{if } x_{[\hat{y}.u_i, \hat{y}.v_i]} = \text{ tire maker} \\ & \hat{y}.t_{(i-1)}, \hat{y}.t_i = \perp, \texttt{ORG} \\ 0 & \text{otherwise} \end{cases}$$

This feature is triggered if the labels of the $(i-1)$-th and the i-th segments are "$\perp$, ORG", and the text of the i-th segment is "tire maker". Our segment-based features are described as follows:

Gazetteer features Entity type of each segment based on matching a number of gazetteers including persons, countries, cities and organizations.

Case features Whether a segment's words are initial-capitalized, all lower cased, or mixture.

Contextual features Unigrams and bigrams of the text and part-of-speech tags in a segment's contextual window of size 2.

Parsing-based features Features derived from constituent parsing trees, including (a) the phrase type of the lowest common ancestor of the tokens contained in the segment, (b) the depth of the lowest common ancestor, (c) a binary feature indicating if the segment is a base phrase or a suffix of a base phrase, and (d) the head words of the segment and its neighbor phrases.

In addition, we convert each triple $\langle u_i, v_i, t_i \rangle$ to BILOU tags for the tokens it contains to implement token-based features. The token-based men-

tion features and local relation features are identical to those of our pipelined system (Section 2.2).

4.2 Global Entity Mention Features

By virtue of the efficient inexact search, we are able to use arbitrary features from the entire structure of $\hat{y}$ to capture long-distance dependencies. The following features between related entity mentions are extracted once a new segment is appended during decoding.

Coreference consistency Coreferential entity mentions should be assigned the same entity type. We determine high-recall coreference links between two segments in the same sentence using some simple heuristic rules:

- Two segments exactly or partially string match.
- A pronoun (e.g., "*their*","*it*") refers to previous entity mentions. For example, in "***they** have no insurance on **their** cars*", "*they*" and "*their*" should have the same entity type.
- A relative pronoun (e.g., "*which*","*that*", and "*who*") refers to the noun phrase it modifies in the parsing tree. For example, in "*the starting kicker is **nikita kargalskiy**, **who** may be 5,000 miles from his hometown*", "*nikita kargalskiy*" and "*who*" should both be labeled as persons.

Then we encode a global feature to check whether two coreferential segments share the same entity type. This feature is particularly effective for pronouns because their contexts alone are often not informative.

Neighbor coherence Neighboring entity mentions tend to have coherent entity types. For example, in "***Barbara Starr** was reporting from the **Pentagon***", "*Barbara Starr*" and "*Pentagon*" are connected by a dependency link *prep_from* and thus they are unlikely to be a pair of PER mentions. Two types of neighbor are considered: (i) the first entity mention before the current segment, and (ii) the segment which is connected by a single word or a dependency link with the current segment. We take the entity types of the two segments and the linkage together as a global feature. For instance, "PER *prep_from* PER" is a feature for the above example when "*Barbara Starr*" and "*Pentagon*" are both labeled as PER mentions.

Part-of-whole consistency If an entity mention is semantically part of another mention (connected by a *prep_of* dependency link), they should be assigned the same entity type. For example, in "***some** of Iraq's **exiles***", "*some*" and "*exiles*" are both PER mentions; in "***one** of the town's two meat-packing **plants***", "*one*" and "*plants*" are both FAC mentions; in "*the **rest** of **America***", "*rest*" and "*America*" are both GPE mentions.

4.3 Global Relation Features

Relation arcs can also share inter-dependencies or obey soft constraints. We extract the following relation-centric global features when a new relation hypothesis is made during decoding.

Role coherence If an entity mention is involved in multiple relations with the same type, then its roles should be coherent. For example, a PER mention is unlikely to have more than one employer. However, a GPE mention can be a physical location for multiple entity mentions. We combine the relation type and the entity mention's argument roles as a global feature, as shown in Figure 5a.

Triangle constraint Multiple entity mentions are unlikely to be fully connected with the same relation type. We use a negative feature to penalize any configuration that contains this type of structure. An example is shown in Figure 5b.

Inter-dependent compatibility If two entity mentions are connected by a dependency link, they tend to have compatible relations with other entities. For example, in Figure 5c, the *conj_and* dependency link between "*Somalia*" and "*Kosovo*" indicates they may share the same relation type with the third entity mention "*forces*".

Neighbor coherence Similar to the entity mention neighbor coherence feature, we also combine the types of two neighbor relations in the same sentence as a bigram feature.

5 Experiments

5.1 Data and Scoring Metric

Most previous work on ACE relation extraction has reported results on ACE'04 data set. As we will show later in our experiments, ACE'05 made significant improvement on both relation type definition and annotation quality. Therefore we present the overall performance on ACE'05 data. We removed two small subsets in informal genres - *cts* and *un*, and then randomly split the remaining 511 documents into 3 parts: 351 for training, 80 for development, and the rest 80 for blind test. In order to compare with state-of-the-art we also performed the same 5-fold cross-validation on *bnews* and *nwire* subsets of ACE'04 corpus as in previous work. The statistics of these data sets

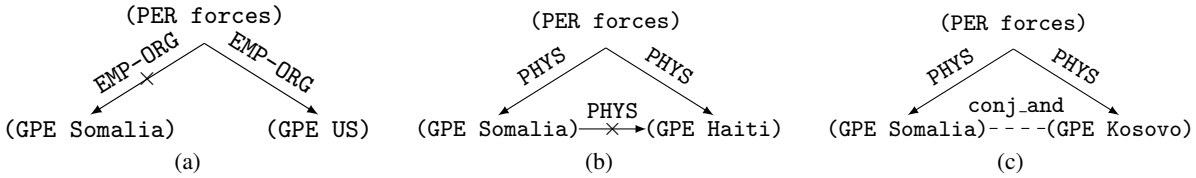
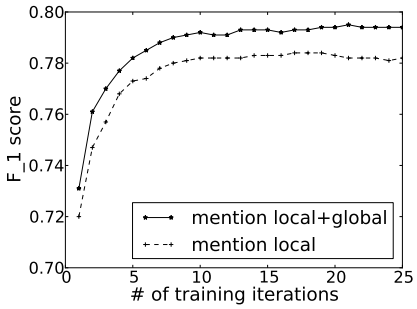
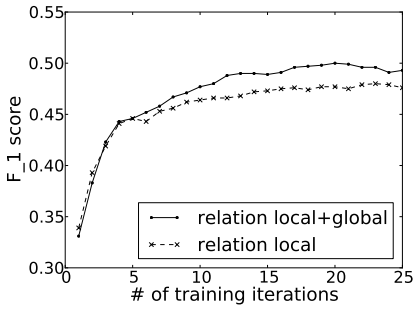


Figure 5: Examples of Global Relation Features.



(a) Entity Mention Performance



(b) Relation Performance

Figure 6: Learning Curves on Development Set.

are summarized in Table 1. We ran the Stanford CoreNLP toolkit[5] to automatically recover the true cases for lowercased documents.

Data Set		# sentences	# mentions	# relations
ACE'05	Train	7,273	26,470	4,779
	Dev	1,765	6,421	1,179
	Test	1,535	5,476	1,147
ACE'04		6,789	22,740	4,368

Table 1: Data Sets.

We use the standard F_1 measure to evaluate the performance of entity mention extraction and relation extraction. An entity mention is considered correct if its entity type is correct and the offsets of its mention head are correct. A relation mention is considered correct if its relation type is correct, and the head offsets of two entity mention arguments are both correct. As in Chan and Roth (2011), we excluded the DISC relation type, and removed relations in the system output which are implicitly correct via coreference links for fair comparison. Furthermore, we combine these two criteria to evaluate the performance of end-to-end entity mention and relation extraction.

[5]http://nlp.stanford.edu/software/corenlp.shtml

5.2 Development Results

In general a larger beam size can yield better performance but increase training and decoding time. As a tradeoff, we set the beam size as 8 throughout the experiments. Figure 6 shows the learning curves on the development set, and compares the performance with and without global features. From these figures we can clearly see that global features consistently improve the extraction performance of both tasks. We set the number of training iterations as 22 based on these curves.

5.3 Overall Performance

Table 2 shows the overall performance of various methods on the ACE'05 test data. We compare our proposed method (Joint w/ Global) with the pipelined system (Pipeline), the joint model with only local features (Joint w/ Local), and two human annotators who annotated 73 documents in ACE'05 corpus.

We can see that our approach significantly outperforms the pipelined approach for both tasks. As a real example, for the partial sentence "*a marcher from Florida*" from the test data, the pipelined approach failed to identify "*marcher*" as a PER mention, and thus missed the GEN-AFF relation between "*marcher*" and "*Florida*". Our joint model correctly identified the entity mentions and their relation. Figure 7 shows the details when the joint model is applied to this sentence. At the token "*marcher*", the top hypothesis in the beam is "$\langle \perp, \perp \rangle$", while the correct one is ranked second best. After the decoder processes the token "*Florida*", the correct hypothesis is promoted to the top in the beam by the *Neighbor Coherence* features for PER-GPE pair. Furthermore, after

Model	Entity Mention (%)			Relation (%)			Entity Mention + Relation (%)		
Score	P	R	F_1	P	R	F_1	P	R	F_1
Pipeline	83.2	73.6	78.1	67.5	39.4	49.8	65.1	38.1	48.0
Joint w/ Local	84.5	76.0	80.0	68.4	40.1	50.6	65.3	38.3	48.3
Joint w/ Global	85.2	76.9	**80.8**	68.9	41.9	**52.1**	65.4	39.8	**49.5**
Annotator 1	91.8	89.9	90.9	71.9	69.0	70.4	69.5	66.7	68.1
Annotator 2	88.7	88.3	88.5	65.2	63.6	64.4	61.8	60.2	61.0
Inter-Agreement	85.8	87.3	86.5	55.4	54.7	55.0	52.3	51.6	51.9

Table 2: Overall performance on ACE'05 corpus.

steps	hypotheses	rank
(a)	$\langle \texttt{a}_\perp \ \texttt{marcher}_\perp \rangle$	1
	$\langle \texttt{a}_\perp \ \texttt{marcher}_{\texttt{PER}} \rangle$	2
(b)	$\langle \texttt{a}_\perp \ \texttt{marcher}_\perp \ \ \texttt{from}_\perp \rangle$	1
	$\langle \texttt{a}_\perp \ \texttt{marcher}_{\texttt{PER}} \ \texttt{from}_\perp \rangle$	4
(c)	$\langle \texttt{a}_\perp \ \texttt{marcher}_{\texttt{PER}} \ \texttt{from}_\perp \ \ \texttt{Florida}_{\texttt{GPE}} \rangle$	1
	$\langle \texttt{a}_\perp \ \texttt{marcher}_\perp \ \ \texttt{from}_\perp \ \ \texttt{Florida}_{\texttt{GPE}} \rangle$	2
(d)	$\langle \texttt{a}_\perp \ \overset{\texttt{GEN-AFF}}{\texttt{marcher}_{\texttt{PER}}} \ \texttt{from}_\perp \ \ \texttt{Florida}_{\texttt{GPE}} \rangle$	1
	$\langle \texttt{a}_\perp \ \texttt{marcher}_\perp \ \ \texttt{from}_\perp \ \ \texttt{Florida}_{\texttt{GPE}} \rangle$	4

Figure 7: Two competing hypotheses for "*a marcher from Florida*" during decoding.

linking the two mentions by GEN-AFF relation, the ranking of the incorrect hypothesis "$\langle \perp, \perp \rangle$" is dropped to the 4-th place in the beam, resulting in a large margin from the correct hypothesis.

The human F_1 score on end-to-end relation extraction is only about 70%, which indicates it is a very challenging task. Furthermore, the F_1 score of the inter-annotator agreement is 51.9%, which is only 2.4% above that of our proposed method.

Compared to human annotators, the bottleneck of automatic approaches is the low recall of relation extraction. Among the 631 remaining missing relations, 318 (50.3%) of them were caused by missing entity mention arguments. A lot of nominal mention heads rarely appear in the training data, such as persons ("supremo", "shepherd", "oligarchs", "rich"), geo-political entity mentions ("stateside"), facilities ("roadblocks", "cells"), weapons ("sim lant", "nukes") and vehicles ("prams"). In addition, relations are often implicitly expressed in a variety of forms. Some examples are as follows:

- "*Rice has been chosen by President Bush to become the new Secretary of State*" indicates "Rice" has a PER-SOC relation with "Bush".
- "*U.S. troops are now knocking on the door of Baghdad*" indicates "troops" has a PHYS relation with "Baghdad".
- "*Russia and France sent planes to Baghdad*" indicates "Russia" and "France" are involved in an ART relation with "planes" as owners.

In addition to contextual features, deeper semantic knowledge is required to capture such implicit semantic relations.

5.4 Comparison with State-of-the-art

Table 3 compares the performance on ACE'04 corpus. For entity mention extraction, our joint model achieved 79.7% on 5-fold cross-validation, which is comparable with the best F_1 score 79.2% reported by (Florian et al., 2006) on single-fold. However, Florian et al. (2006) used some gazetteers and the output of other Information Extraction (IE) models as additional features, which provided significant gains ((Florian et al., 2004)). Since these gazetteers, additional data sets and external IE models are all not publicly available, it is not fair to directly compare our joint model with their results.

For end-to-end entity mention and relation extraction, both the joint approach and the pipelined baseline outperform the best results reported by (Chan and Roth, 2011) under the same setting.

6 Related Work

Entity mention extraction (e.g., (Florian et al., 2004; Florian et al., 2006; Florian et al., 2010; Zitouni and Florian, 2008; Ohta et al., 2012)) and relation extraction (e.g., (Reichartz et al., 2009; Sun et al., 2011; Jiang and Zhai, 2007; Bunescu and Mooney, 2005; Zhao and Grishman, 2005; Culotta and Sorensen, 2004; Zhou et al., 2007; Qian and Zhou, 2010; Qian et al., 2008; Chan and Roth, 2011; Plank and Moschitti, 2013)) have drawn much attention in recent years but were

Model	Entity Mention (%)			Relation (%)			Entity Mention + Relation (%)		
Score	P	R	F_1	P	R	F_1	P	R	F_1
Chan and Roth (2011)		-		42.9	38.9	40.8		-	
Pipeline	81.5	74.1	77.6	62.5	36.4	46.0	58.4	33.9	42.9
Joint w/ Local	82.7	75.2	78.8	64.2	37.0	46.9	60.3	34.8	44.1
Joint w/ Global	83.5	76.2	**79.7**	64.7	38.5	**48.3**	60.8	36.1	**45.3**

Table 3: 5-fold cross-validation on ACE'04 corpus. Bolded scores indicate highly statistical significant improvement as measured by paired t-test ($p < 0.01$)

usually studied separately. Most relation extraction work assumed that entity mention boundaries and/or types were given. Chan and Roth (2011) reported the best results using predicted entity mentions.

Some previous work used relations and entity mentions to enhance each other in joint inference frameworks, including re-ranking (Ji and Grishman, 2005), Integer Linear Programming (ILP) (Roth and Yih, 2004; Roth and Yih, 2007; Yang and Cardie, 2013), and Card-pyramid Parsing (Kate and Mooney, 2010). All these work noted the advantage of exploiting cross-component interactions and richer knowledge. However, they relied on models separately learned for each subtask. As a key difference, our approach jointly extracts entity mentions and relations using a single model, in which arbitrary soft constraints can be easily incorporated. Some other work applied probabilistic graphical models for joint extraction (e.g., (Singh et al., 2013; Yu and Lam, 2010)). By contrast, our work employs an efficient joint search algorithm without modeling joint distribution over numerous variables, therefore it is more flexible and computationally simpler. In addition, (Singh et al., 2013) used gold-standard mention boundaries.

Our previous work (Li et al., 2013) used structured perceptron with token-based decoder to jointly predict event triggers and arguments based on the assumption that entity mentions and other argument candidates are given as part of the input. In this paper, we solve a more challenging problem: take raw texts as input and identify the boundaries, types of entity mentions and relations all together in a single model. Sarawagi and Cohen (2004) proposed a segment-based CRFs model for name tagging. Zhang and Clark (2008) used a segment-based decoder for word segmentation and pos tagging. We extended the similar idea to our end-to-end task by incrementally predicting relations along with entity mention segments.

7 Conclusions and Future Work

In this paper we introduced a new architecture for more powerful end-to-end entity mention and relation extraction. For the first time, we addressed this challenging task by an incremental beam-search algorithm in conjunction with structured perceptron. While detecting mention boundaries jointly with other components raises the challenge of synchronizing multiple assignments in the same beam, a simple yet effective segment-based decoder is adopted to solve this problem. More importantly, we exploited a set of global features based on linguistic and logical properties of the two tasks to predict more coherent structures. Experiments demonstrated our approach significantly outperformed pipelined approaches for both tasks and dramatically advanced state-of-the-art.

In future work, we plan to explore more soft and hard constraints to reduce search space as well as improve accuracy. In addition, we aim to incorporate other IE components such as event extraction into the joint model.

Acknowledgments

We thank the three anonymous reviewers for their insightful comments. This work was supported by the U.S. Army Research Laboratory under Cooperative Agreement No. W911NF-09-2-0053 (NSCTA), U.S. NSF CAREER Award under Grant IIS-0953149, U.S. DARPA Award No. FA8750-13-2-0041 in the Deep Exploration and Filtering of Text (DEFT) Program, IBM Faculty Award, Google Research Award and RPI faculty start-up grant. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Razvan C. Bunescu and Raymond J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proc. HLT/EMNLP*, pages 724–731.

Yee Seng Chan and Dan Roth. 2010. Exploiting background knowledge for relation extraction. In *Proc. COLING*, pages 152–160.

Yee Seng Chan and Dan Roth. 2011. Exploiting syntactico-semantic structures for relation extraction. In *Proc. ACL*, pages 551–560.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proc. ACL*, pages 111–118.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. EMNLP*, pages 1–8.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proc. ACL*, pages 423–429.

Radu Florian, Hany Hassan, Abraham Ittycheriah, Hongyan Jing, Nanda Kambhatla, Xiaoqiang Luo, Nicolas Nicolov, and Salim Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *Proc. HLT-NAACL*, pages 1–8.

Radu Florian, Hongyan Jing, Nanda Kambhatla, and Imed Zitouni. 2006. Factorizing complex models: A case study in mention detection. In *Proc. ACL*.

Radu Florian, John F. Pitrelli, Salim Roukos, and Imed Zitouni. 2010. Improving mention detection robustness to noisy input. In *Proc. EMNLP*, pages 335–345.

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *ACL*, pages 1077–1086.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proc. HLT-NAACL*, pages 142–151.

Heng Ji and Ralph Grishman. 2005. Improving name tagging by reference resolution and relation detection. In *Proc. ACL*, pages 411–418.

Jing Jiang and ChengXiang Zhai. 2007. A systematic exploration of the feature space for relation extraction. In *Proc. HLT-NAACL*.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for information extraction. In *Proc. ACL*, pages 178–181.

Rohit J. Kate and Raymond Mooney. 2010. Joint entity and relation extraction using card-pyramid parsing. In *Proc. ACL*, pages 203–212.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*, pages 282–289.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proc. ACL*, pages 73–82.

Marie-Catherine De Marneffe, Bill Maccartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proc. LREC*, pages 449,454.

Tomoko Ohta, Sampo Pyysalo, Jun'ichi Tsujii, and Sophia Ananiadou. 2012. Open-domain anatomical entity mention detection. In *Proc. ACL Workshop on Detecting Structure in Scholarly Discourse*, pages 27–36.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *Proc. ACL*, pages 1498–1507.

Longhua Qian and Guodong Zhou. 2010. Clustering-based stratified seed sampling for semi-supervised relation classification. In *Proc. EMNLP*, pages 346–355.

Longhua Qian, Guodong Zhou, Fang Kong, Qiaoming Zhu, and Peide Qian. 2008. Exploiting constituent dependencies for tree kernel-based semantic relation extraction. In *Proc. COLING*, pages 697–704.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proc. CONLL*, pages 147–155.

Frank Reichartz, Hannes Korte, and Gerhard Paass. 2009. Composite kernels for relation extraction. In *Proc. ACL-IJCNLP (Short Papers)*, pages 365–368.

Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proc. CoNLL*.

Dan Roth and Wen-tau Yih. 2007. Global inference for entity and relation identification via a lin- ear programming formulation. In *Introduction to Statistical Relational Learning*. MIT.

Sunita Sarawagi and William W. Cohen. 2004. Semimarkov conditional random fields for information extraction. In *Proc. NIPS*.

Sameer Singh, Sebastian Riedel, Brian Martin, Jiaping Zheng, and Andrew McCallum. 2013. Joint inference of entities, relations, and coreference. In *Proc. CIKM Workshop on Automated Knowledge Base Construction*.

Ang Sun, Ralph Grishman, and Satoshi Sekine. 2011. Semi-supervised relation extraction with large-scale word clustering. In *Proc. ACL*, pages 521–529.

Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In *Proc. ACL*, pages 1640–1649.

Xiaofeng Yu and Wai Lam. 2010. Jointly identifying entities and extracting relations in encyclopedia text via a graphical model approach. In *Proc. COLING (Posters)*, pages 1399–1407.

Yue Zhang and Stephen Clark. 2008. Joint word segmentation and pos tagging using a single perceptron. In *Proc. ACL*, pages 1147–1157.

Shubin Zhao and Ralph Grishman. 2005. Extracting relations with integrated information using kernel methods. In *Proc. ACL*, pages 419–426.

Guodong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proc. ACL*, pages 427–434.

Guodong Zhou, Min Zhang, Dong-Hong Ji, and Qiaoming Zhu. 2007. Tree kernel-based relation extraction with context-sensitive structured parse tree information. In *Proc. EMNLP-CoNLL*, pages 728–736.

Imed Zitouni and Radu Florian. 2008. Mention detection crossing the language barrier. In *Proc. EMNLP*, pages 600–609.

That's Not What I Meant! Using Parsers to Avoid Structural Ambiguities in Generated Text

Manjuan Duan and **Michael White**
Department of Linguistics
The Ohio State University
Columbus, OH 43210, USA
{duan,mwhite}@ling.osu.edu

Abstract

We investigate whether parsers can be used for self-monitoring in surface realization in order to avoid egregious errors involving "vicious" ambiguities, namely those where the intended interpretation fails to be considerably more likely than alternative ones. Using parse accuracy in a simple reranking strategy for self-monitoring, we find that with a state-of-the-art averaged perceptron realization ranking model, BLEU scores cannot be improved with any of the well-known Treebank parsers we tested, since these parsers too often make errors that human readers would be unlikely to make. However, by using an SVM ranker to combine the realizer's model score together with features from multiple parsers, including ones designed to make the ranker more robust to parsing mistakes, we show that significant increases in BLEU scores can be achieved. Moreover, via a targeted manual analysis, we demonstrate that the SVM reranker frequently manages to avoid vicious ambiguities, while its ranking errors tend to affect fluency much more often than adequacy.

1 Introduction

Rajkumar & White (2011; 2012) have recently shown that some rather egregious surface realization errors—in the sense that the reader would likely end up with the wrong interpretation—can be avoided by making use of features inspired by psycholinguistics research together with an otherwise state-of-the-art averaged perceptron realization ranking model (White and Rajkumar, 2009), as reviewed in the next section. However, one is apt to wonder: could one use a parser to check whether the intended interpretation is easy to recover, either as an alternative or to catch additional mistakes? Doing so would be tantamount to self-monitoring in Levelt's (1989) model of language production.

Neumann & van Noord (1992) pursued the idea of self-monitoring for generation in early work with reversible grammars. As Neumann & van Noord observed, a simple, brute-force way to generate unambiguous sentences is to enumerate possible realizations of an input logical form, then to parse each realization to see how many interpretations it has, keeping only those that have a single reading; they then went on to devise a more efficient method of using self-monitoring to avoid generating ambiguous sentences, targeted to the ambiguous portion of the output. We might question, however, whether it is really possible to avoid ambiguity entirely in the general case, since Abney (1996) and others have argued that nearly every sentence is potentially ambiguous, though we (as human comprehenders) may not notice the ambiguities if they are unlikely. Taking up this issue, Khan et al. (2008)—building on Chantree et al.'s (2006) approach to identifying "innocuous" ambiguities—conducted several experiments to test whether ambiguity could be balanced against length or fluency in the context of generating referring expressions involving coordinate structures. Though Khan et al.'s study was limited to this one kind of structural ambiguity, they do observe that generating the brief variants when the intended interpretation is clear instantiates Van Deemter's (2004) general strategy of only avoiding **vicious ambiguities**—that is, ambiguities where the intended interpretation fails to be considerably more likely than any other distractor interpretations—rather than trying to avoid all ambiguities.

In this paper, we investigate whether Neumann & van Noord's brute-force strategy for avoid-

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 413–423,
Baltimore, Maryland, USA, June 23-25 2014.

ing ambiguities in surface realization can be updated to only avoid vicious ambiguities, extending (and revising) Van Deemter's general strategy to all kinds of structural ambiguity, not just the one investigated by Khan et al. To do so—in a nutshell—we enumerate an n-best list of realizations and rerank them if necessary to avoid vicious ambiguities, as determined by one or more automatic parsers. A potential obstacle, of course, is that automatic parsers may not be sufficiently representative of human readers, insofar as errors that a parser makes may not be problematic for human comprehension; moreover, parsers are rarely successful in fully recovering the intended interpretation for sentences of moderate length, even with carefully edited news text. Consequently, we examine two reranking strategies, one a simple baseline approach and the other using an SVM reranker (Joachims, 2002).

Our simple reranking strategy for self-monitoring is to rerank the realizer's n-best list by parse accuracy, preserving the original order in case of ties. In this way, if there is a realization in the n-best list that can be parsed more accurately than the top-ranked realization—even if the intended interpretation cannot be recovered with 100% accuracy—it will become the preferred output of the combined realization-with-self-monitoring system. With this simple reranking strategy and each of three different Treebank parsers, we find that it is possible to improve BLEU scores on Penn Treebank development data with White & Rajkumar's (2011; 2012) baseline generative model, but not with their averaged perceptron model. In inspecting the results of reranking with this strategy, we observe that while it does sometimes succeed in avoiding egregious errors involving vicious ambiguities, common parsing mistakes such as PP-attachment errors lead to unnecessarily sacrificing conciseness or fluency in order to avoid ambiguities that would be easily tolerated by human readers. Therefore, to develop a more nuanced self-monitoring reranker that is more robust to such parsing mistakes, we trained an SVM using dependency precision and recall features for all three parses, their n-best parsing results, and per-label precision and recall for each type of dependency, together with the realizer's normalized perceptron model score as a feature. With the SVM reranker, we obtain a significant improvement in BLEU scores over White & Rajkumar's averaged perceptron model on both development and test data. Additionally, in a targeted manual analysis, we find that in cases where the SVM reranker improves the BLEU score, improvements to fluency and adequacy are roughly balanced, while in cases where the BLEU score goes down, it is mostly fluency that is made worse (with reranking yielding an acceptable paraphrase roughly one third of the time in both cases).

The paper is structured as follows. In Section 2, we review the realization ranking models that serve as a starting point for the paper. In Section 3, we report on our experiments with the simple reranking strategy, including a discussion of the ways in which this method typically fails. In Section 4, we describe how we trained an SVM reranker and report our results using BLEU scores (Papineni et al., 2002). In Section 5, we present a targeted manual analysis of the development set sentences with the greatest change in BLEU scores, discussing both successes and errors. In Section 6, we briefly review related work on broad coverage surface realization. Finally, in Section 7, we sum up and discuss opportunities for future work in this direction.

2 Background

We use the OpenCCG[1] surface realizer for the experiments reported in this paper. The OpenCCG realizer generates surface strings for input semantic dependency graphs (or logical forms) using a chart-based algorithm (White, 2006) for Combinatory Categorial Grammar (Steedman, 2000) together with a "hypertagger" for probabilistically assigning lexical categories to lexical predicates in the input (Espinosa et al., 2008). An example input appears in Figure 1. In the figure, nodes correspond to discourse referents labeled with lexical predicates, and dependency relations between nodes encode argument structure (gold standard CCG lexical categories are also shown); note that semantically empty function words such as infinitival-*to* are missing. The grammar is extracted from a version of the CCGbank (Hockenmaier and Steedman, 2007) enhanced for realization; the enhancements include: better analyses of punctuation (White and Rajkumar, 2008); less error prone handling of named entities (Rajkumar et al., 2009); re-inserting quotes into the CCGbank;

[1] `http://openccg.sf.net`

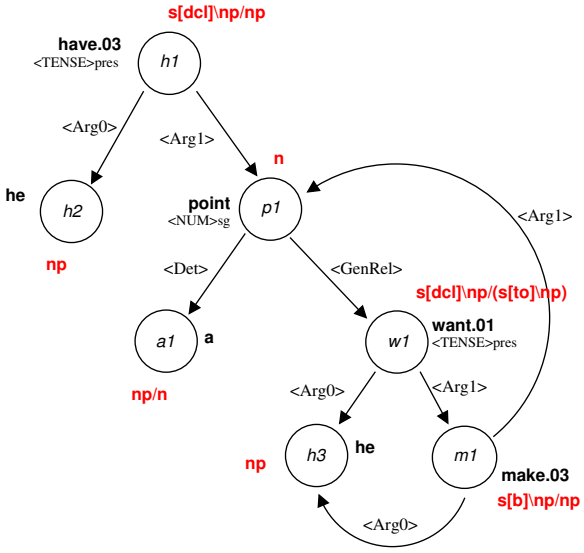


Figure 1: Example OpenCCG semantic dependency input for *he has a point he wants to make*, with gold standard lexical categories for each node

and assignment of consistent semantic roles across diathesis alternations (Boxwell and White, 2008), using PropBank (Palmer et al., 2005).

To select preferred outputs from the chart, we use White & Rajkumar's (2009; 2012) realization ranking model, recently augmented with a large-scale 5-gram model based on the Gigaword corpus. The ranking model makes choices addressing all three interrelated sub-tasks traditionally considered part of the surface realization task in natural language generation research (Reiter and Dale, 2000; Reiter, 2010): inflecting lemmas with grammatical word forms, inserting function words and linearizing the words in a grammatical and natural order. The model takes as its starting point two probabilistic models of syntax that have been developed for CCG parsing, Hockenmaier & Steedman's (2002) generative model and Clark & Curran's (2007) normal-form model. Using the averaged perceptron algorithm (Collins, 2002), White & Rajkumar (2009) trained a structured prediction ranking model to combine these existing syntactic models with several n-gram language models. This model improved upon the state-of-the-art in terms of automatic evaluation scores on held-out test data, but nevertheless an error analysis revealed a surprising number of word order, function word and inflection errors. For each kind of error, subsequent work investigated the utility of employing more linguistically motivated features to improve the ranking model.

To improve word ordering decisions, White & Rajkumar (2012) demonstrated that incorporating a feature into the ranker inspired by Gibson's (2000) dependency locality theory can deliver statistically significant improvements in automatic evaluation scores, better match the distributional characteristics of sentence orderings, and significantly reduce the number of serious ordering errors (some involving vicious ambiguities) as confirmed by a targeted human evaluation. Supporting Gibson's theory, comprehension and corpus studies have found that the tendency to **minimize dependency length** has a strong influence on constituent ordering choices; see Temperley (2007) and Gildea and Temperley (2010) for an overview.

Table 1 shows examples from White and Rajkumar (2012) of how the dependency length feature (DEPLEN) affects the OpenCCG realizer's output even in comparison to a model (DEPORD) with a rich set of discriminative syntactic and dependency ordering features, but no features directly targeting relative weight. In wsj_0015.7, the dependency length model produces an exact match, while the DEPORD model fails to shift the short temporal adverbial *next year* next to the verb, leaving a confusingly repetitive *this year next year* at the end of the sentence. Note how shifting *next year* from its canonical VP-final position to appear next to the verb shortens its dependency length considerably, while barely lengthening the dependency to *based on*; at the same time, it avoids ambiguity in what *next year* is modifying. In wsj_0020.1 we see the reverse case: the dependency length model produces a nearly exact match with just an equally acceptable inversion of *closely watching*, keeping the direct object in its canonical position. By contrast, the DEPORD model mistakenly shifts the direct object *South Korea, Taiwan and Saudia Arabia* to the end of the sentence where it is difficult to understand following two very long intervening phrases.

With function words, Rajkumar and White (2011) showed that they could improve upon the earlier model's predictions for when to employ *that*-complementizers using features inspired by Jaeger's (2010) work on using the principle of **uniform information density**, which holds that human language use tends to keep information density relatively constant in order to optimize communicative efficiency. In news text, com-

wsj_0015.7	the exact amount of the refund will be determined **next year** based on actual collections made until Dec. 31 of this year .
DEPLEN	[same]
DEPORD	the exact amount of the refund will be determined based on actual collections made until Dec. 31 of this year *next year* .
wsj_0020.1	the U.S. , claiming some success in its trade diplomacy , removed South Korea , Taiwan and Saudi Arabia from a list of countries it is closely watching for allegedly failing to honor U.S. patents , copyrights and other intellectual-property rights .
DEPLEN	the U.S. claiming some success in its trade diplomacy , removed **South Korea , Taiwan and Saudi Arabia** from a list of countries it is *watching closely* for allegedly failing to honor U.S. patents , copyrights and other intellectual-property rights .
DEPORD	the U.S. removed from a list of countries it is *watching closely* for allegedly failing to honor U.S. patents , copyrights and other intellectual-property rights , claiming some success in its trade diplomacy , *South Korea , Taiwan and Saudi Arabia* .

Table 1: Examples of realized output for full models with and without the dependency length feature (White and Rajkumar, 2012)

plementizers are left out two times out of three, but in some cases the presence of *that* is crucial to the interpretation. Generally, inserting a complementizer makes the onset of a complement clause more predictable, and thus less information dense, thereby avoiding a potential spike in information density that is associated with comprehension difficulty. Rajkumar & White's experiments confirmed the efficacy of the features based on Jaeger's work, including information density–based features, in a local classification model.[2] Their experiments also showed that the improvements in prediction accuracy apply to cases in which the presence of a *that*-complementizer arguably makes a substantial difference to fluency or intelligiblity. For example, in (1), the presence of *that* avoids a local ambiguity, helping the reader to understand that *for the second month in a row* modifies the reporting of the shortage; without *that*, it is very easy to mis-parse the sentence as having *for the second month in a row* modifying the saying event.

(1) He said that/∅? for the second month in a row, food processors reported a shortage of nonfat dry milk. (PTB WSJ0036.61)

Finally, to reduce the number of subject-verb agreement errors, Rajkumar and White (2010) extended the earlier model with features enabling it to make correct verb form choices in sentences involving complex coordinate constructions and with expressions such as *a lot of* where the correct choice is not determined solely by the head noun. They also improved animacy agreement with relativizers, reducing the number of errors where *that* or *which* was chosen to modify an animate noun rather than *who* or *whom* (and vice-versa), while also allowing both choices where corpus evidence was mixed.

[2]Note that the features from the local classification model for *that*-complementizer choice have not yet been incorporated into OpenCCG's global realization ranking model, and thus do not inform the baseline realization choices in this work.

3 Simple Reranking

3.1 Methods

We ran two OpenCCG surface realization models on the CCGbank dev set (derived from Section 00 of the Penn Treebank) and obtained n-best ($n = 10$) realizations. The first one is the baseline generative model (hereafter, generative model) used in training the averaged perceptron model. This model ranks realizations using the product of the Hockenmaier syntax model, n-gram models over words, POS tags and supertags in the training sections of the CCGbank, and the large-scale 5-gram model from Gigaword. The second one is the averaged perceptron model (hereafter, perceptron model), which uses all the features reviewed in Section 2. In order to experiment with multiple parsers, we used the Stanford dependencies (de Marneffe et al., 2006), obtaining gold dependencies from the gold-standard PTB parses and automatic dependencies from the automatic parses of each realization. Using dependencies allowed us to measure parse accuracy independently of word order. We chose the Berkeley parser (Petrov et al., 2006), Brown parser (Charniak and Johnson, 2005) and Stanford parser (Klein and Manning, 2003) to parse the realizations generated by the

	Berkeley	Brown	Stanford
No reranking	87.93	87.93	87.93
Labeled	87.77	87.87	87.12
Unlabeled	87.90	**87.97**	86.97

Table 2: Devset BLEU scores for simple ranking on top of n-best perceptron model realizations

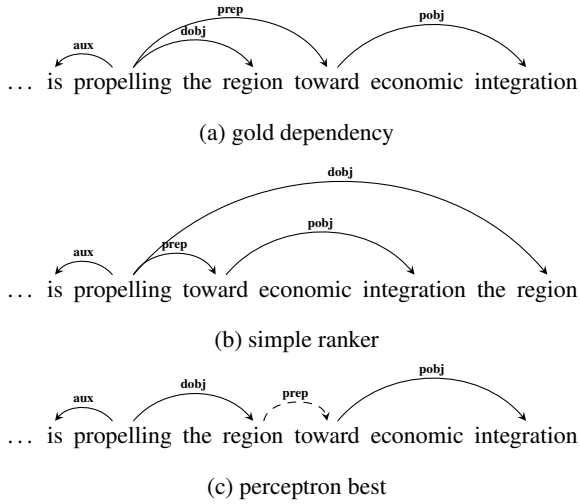


Figure 2: Example parsing mistake in PP-attachment (wsj_0043.1)

two realization models and calculated precision, recall and F_1 of the dependencies for each realization by comparing them with the gold dependencies. We then ranked the realizations by their F_1 score of parse accuracy, keeping the original ranking in case of ties. We also tried using unlabeled (and unordered) dependencies, in order to possibly make better use of parses that were close to being correct. In this setting, as long as the right pair of tokens occur in a dependency relation, it was counted as a correctly recovered dependency.

3.2 Results

Simple ranking with the Berkeley parser of the generative model's n-best realizations raised the BLEU score from 85.55 to 86.07, well below the averaged perceptron model's BLEU score of 87.93. However, as shown in Table 2, none of the parsers yielded significant improvements on the top of the perceptron model.

Inspecting the results of simple ranking revealed that while simple ranking did successfully avoid vicious ambiguities in some cases, parser mistakes with PP-attachments, noun-noun compounds and coordinate structures too often blocked the gold realization from emerging on top. To illustrate, Figure 2 shows an example with a PP-attachment mistake. In the figure, the key gold dependencies of the reference sentence are shown in (a), the dependencies of the realization selected by the simple ranker are shown in (b), and the dependencies of the realization selected by the perceptron ranker (same as gold) appear in (c), with the parsing mistake indicated by the dashed line. The simple ranker ends up choosing (b) as the best realization because it has the most accurate parse compared to the reference sentence, given the mistake with (c).

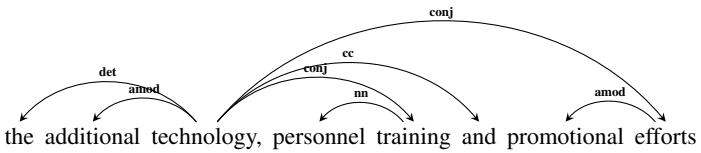
Other common parse errors are illustrated in Figure 3. Here, (b) ends up getting chosen by the simple ranker as the realization with the most accurate parse given the failures in (c), where *the additional technology, personnel training* is mistakenly analyzed as one noun phrase, a reading unlikely to be considered by human readers.

In sum, although simple ranking helps to avoid vicious ambiguity in some cases, the overall results of simple ranking are no better than the perceptron model (according to BLEU, at least), as parse failures that are not reflective of human intepretive tendencies too often lead the ranker to choose dispreferred realizations. As such, we turn now to a more nuanced model for combining the results of multiple parsers in a way that is less sensitive to such parsing mistakes, while also letting the perceptron model have a say in the final ranking.

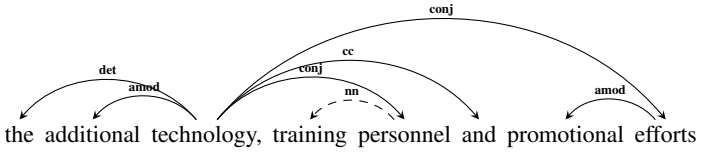
4 Reranking with SVMs

4.1 Methods

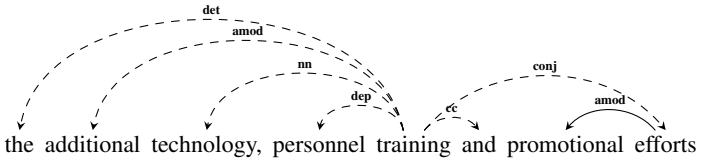
Since different parsers make different errors, we conjectured that dependencies in the intersection of the output of multiple parsers may be more reliable and thus may more reliably reflect human comprehension preferences. Similarly, we conjectured that large differences in the realizer's perceptron model score may more reliably reflect human fluency preferences than small ones, and thus we combined this score with features for parser accuracy in an SVM ranker. Additionally, given that parsers may more reliably recover some kinds of dependencies than others, we included features for each dependency type, so that the SVM ranker might learn how to weight them appropriately. Finally, since the differences among the n-best parses reflect the least certain parsing decisions,



(a) gold dependency



(b) simple ranker



(c) perceptron best

Figure 3: Example parsing mistakes in a noun-noun compound and a coordinate structure (wsj_0085.45)

and thus ones that may require more common sense inference that is easy for humans but not machines, we conjectured that including features from the n-best parses may help to better match human performance. In more detail, we made use of the following feature classes for each candidate realization:

perceptron model score the score from the realizer's model, normalized to [0,1] for the realizations in the n-best list

precision and recall labeled and unlabeled precision and recall for each parser's best parse

per-label precision and recall (*dep*) precision and recall for each type of dependency obtained from each parser's best parse (using zero if not defined for lack of predicted or gold dependencies with a given label)

n-best precision and recall (*nbest*) labeled and unlabeled precision and recall for each parser's top five parses, along with the same features for the most accurate of these parses

In training, we used the BLEU scores of each realization compared with its reference sentence to establish a preference order over pairs of candidate realizations, assuming that the original corpus sentences are generally better than related alternatives, and that BLEU can somewhat reliably predict human preference judgments.

We trained the SVM ranker (Joachims, 2002) with a linear kernel and chose the hyper-parameter c, which tunes the trade-off between training error and margin, with 6-fold cross-validation on the devset. We trained different models to investigate the contribution made by different parsers and different types of features, with the perceptron model score included as a feature in all models. For each parser, we trained a model with its overall precision and recall features, as shown at the top of Table 3. Then we combined these three models to get a new model (Bkl+Brw+St in the table) . Next, to this combined model we separately added (i) the per-label precision and recall features from all the parsers (BBS+dep), and (ii) the n-best features from the parsers (BBS+nbest). The full model (BBS+dep+nbest) includes all the features listed above. Finally, since the Berkeley parser yielded the best results on its own, we also tested models using all the feature classes but only using this parser by itself.

4.2 Results

Table 3 shows the results of different SVM ranking models on the devset. We calculated significance using paired bootstrap resampling (Koehn, 2004).[3] Both the per-label precision & recall fea-

[3]Kudos to Kevin Gimpel for making his implementation available: `http://www.ark.cs.cmu.edu/MT/paired_bootstrap_v13a.tar.gz`

	BLEU	sig.
perceptron baseline	87.93	–
Berkeley	88.45	*
Brown	88.34	
Stanford	88.18	
Bkl+Brw+St	88.44	*
BBS+dep	88.63	**
BBS+nbest	88.60	**
BBS+dep+nbest	**88.73**	**
Bkl+dep	88.63	**
Bkl+nbest	88.48	*
Bkl +dep+nbest	88.68	**

Table 3: Devset results of SVM ranking on top of perceptron model. Significance codes: $**$ for $p < 0.05$, $*$ for $p < 0.1$.

	BLEU	sig.
perceptron baseline	86.94	–
BBS+dep+nbest	**87.64**	**

Table 4: Final test results of SVM ranking on top of perceptron model. Significance codes: $**$ for $p < 0.05$, $*$ for $p < 0.1$.

tures and the n-best parse features contributed to achieving a significant improvement compared to the perceptron model. Somewhat surprisingly, the Berkeley parser did as well as all three parsers using just the overall precision and recall features, but not quite as well using all features. The complete model, BBS+dep+nbest, achieved a BLEU score of 88.73, significantly improving upon the perceptron model ($p < 0.02$). We then confirmed this result on the final test set, Section 23 of the CCGbank, as shown in Table 4 ($p < 0.02$ as well).

5 Analysis and Discussion

5.1 Targeted Manual Analysis

In order to gain a better understanding of the successes and failures of our SVM ranker, we present here a targeted manual analysis of the development set sentences with the greatest change in BLEU scores, carried out by the second author (a native speaker). In this analysis, we consider whether the reranked realization improves upon or detracts from realization quality—in terms of adequacy, fluency, both or neither—along with a linguistic categorization of the differences between the reranked realization and the original top-ranked realization according to the averaged perceptron model. Unlike the broad-based and objective evaluation in terms of BLEU scores presented above, this analysis is narrowly targeted and subjective, though the interested reader is invited to review the complete set of analyzed examples that accompany the paper as a supplement. We leave a more broad-based human evaluation by naive subjects for future work.

Table 5 shows the results of the analysis, both overall and for the most frequent categories of changes. Of the 50 sentences where the BLEU score went up the most, 15 showed an improvement in adequacy (i.e., in conveying the intended meaning), 22 showed an improvement in fluency (with 3 cases also improving adequacy), and 16 yielded no discernible change in fluency or adequacy. By contrast, with the 50 sentences where the BLEU score went down the most, adequacy was only affected 4 times, though fluency was affected 32 times, and 15 remained essentially unchanged.[4] The table also shows that differences in the order of VP constituents usually led to a change in adequacy or fluency, as did ordering changes within NPs, with noun-noun compounds and named entities as the most frequent subcategories of NP-ordering changes. Of the cases where adequacy and fluency were not affected, contractions and subject-verb inversions were the most frequent differences.

Examples of the changes yielded by the SVM ranker appear in Table 6. With wsj_0036.54, the averaged perceptron model selects a realization that regrettably (though amusingly) swaps *purchasing* and *more than 250*—yielding a sentence that suggests that the executives have been purchased!—while the SVM ranker succeeds in ranking the original sentence above all competing realizations. With wsj_0088.25, self-monitoring with the SVM ranker yields a realization nearly identical to the original except for an extra comma, where it is clear that *in public* modifies *do this*; by contrast, in the perceptron-best realization, *in public* mistakenly appears to modify *be disclosed*. With wsj_0041.18, the SVM ranker unfortunately prefers a realization where *presumably* seems to modify *shows* rather than *of two politicians* as

[4]The difference in the distribution of adequacy change, fluency change and no change counts between the two conditions is highly significant statistically ($\chi^2 = 9.3, df = 2, p < 0.01$). In this comparison, items where both fluency and adequacy were affected were counted as adequacy cases.

	±adq	±flu	=eq	±vpord	±npord	±nn	±ne	=vpord	=sbjinv	=cntrc
BLEU wins	15	22	16	10	9	7	3	4	-	11
BLEU losses	4	32	15	8	13	5	5	4	7	-

Table 5: Manual analysis of devset sentences where the SVM ranker achieved the greatest increase/decrease in BLEU scores (50 each of wins/losses) compared to the averaged perceptron baseline model in terms of positive or negative changes in adequacy (±adq), fluency (±flu) or neither (=eq); changes in VP ordering (±vpord), NP ordering (±npord), noun-noun compound ordering (±nn) and named entities (±ne); and neither positive nor negative changes in VP ordering (=vpord), subject-inversion (=sbjinv) and contractions (=cntrc). In all but one case (counted as =eq here), the BLEU wins saw positive changes and the BLEU losses saw negative changes.

wsj_0036.54	the purchasing managers ' report is based on data provided by more than 250 **purchasing** executives .
SVM RANKER	[same]
PERCEP BEST	the purchasing managers ' report is based on data provided by *purchasing* more than 250 executives .
wsj_0088.25	Markey said we could have done this **in public** because so little sensitive information was disclosed , the aide said .
SVM RANKER	Markey said , we could have done this **in public** because so little sensitive information was disclosed , the aide said .
PERCEP BEST	Markey said , we could have done this because so little sensitive information was disclosed *in public* , the aide said .
wsj_0041.18	the screen shows two distorted , unrecognizable photos , **presumably** of two politicians .
SVM RANKER	the screen shows two distorted , unrecognizable photos *presumably* , of two politicians .
PERCEP BEST	[same as original]
wsj_0044.111	" I was dumbfounded " , Mrs. Ward **recalls** .
SVM RANKER	" I was dumbfounded " , *recalls* Mrs. Ward .
PERCEP BEST	[same as original]

Table 6: Examples of devset sentences where the SVM ranker improved adequacy (top), made it worse (middle) or left it the same (bottom)

in the original, which the averaged perceptron model prefers. Finally, wsj_0044.111 is an example where a subject-inversion makes no difference to adequacy or fluency.

5.2 Discussion

The BLEU evaluation and targeted manual analysis together show that the SVM ranker increases the similarity to the original corpus of realizations produced with self-monitoring, often in ways that are crucial for the intended meaning to be apparent to human readers.

A limitation of the experiments reported in this paper is that OpenCCG's input semantic dependency graphs are not the same as the Stanford dependencies used with the Treebank parsers, and thus we have had to rely on the gold parses in the PTB to derive gold dependencies for measuring accuracy of parser dependency recovery. In a realistic application scenario, however, we would need to measure parser accuracy relative to the realizer's input. We initially tried using OpenCCG's parser in a simple ranking approach, but found that it did not improve upon the averaged perceptron model, like the three parsers used subsequently. Given that with the more refined SVM ranker, the Berkeley parser worked nearly as well as all three parsers together using the complete feature set, the prospects for future work on a more realistic scenario using the OpenCCG parser in an SVM ranker for self-monitoring now appear much more promising, either using OpenCCG's reimplementation of Hockenmaier & Steedman's generative CCG model, or using the Berkeley parser trained on OpenCCG's enhanced version of the CCGbank, along the lines of Fowler and Penn (2010).

6 Related Work

Approaches to surface realization have been developed for LFG, HPSG, and TAG, in addition to CCG, and recently statistical dependency-based approaches have been developed as well; see the report from the first surface realization shared

task (Belz et al., 2010; Belz et al., 2011) for an overview. To our knowledge, however, a comprehensive investigation of avoiding vicious structural ambiguities with broad coverage statistical parsers has not been previously explored. As our SVM ranking model does not make use of CCG-specific features, we would expect our self-monitoring method to be equally applicable to realizers using other frameworks.

7 Conclusion

In this paper, we have shown that while using parse accuracy in a simple reranking strategy for self-monitoring fails to improve BLEU scores over a state-of-the-art averaged perceptron realization ranking model, it is possible to significantly increase BLEU scores using an SVM ranker that combines the realizer's model score together with features from multiple parsers, including ones designed to make the ranker more robust to parsing mistakes that human readers would be unlikely to make. Additionally, via a targeted manual analysis, we showed that the SVM reranker frequently manages to avoid egregious errors involving "vicious" ambiguities, of the kind that would mislead human readers as to the intended meaning.

As noted in Reiter's (2010) survey, many NLG systems use surface realizers as off-the-shelf components. In this paper, we have focused on broad coverage surface realization using widely-available PTB data—where there are many sentences of varying complexity with gold-standard annotations—following the common assumption that experiments with broad coverage realization are (or eventually will be) relevant for NLG applications. Of course, the kinds of ambiguity that can be problematic in news text may or may not be the same as the ones encountered in particular applications. Moreover, for certain applications (e.g. ones with medical or legal implications), it may be better to err on the side of ambiguity avoidance, even at some expense to fluency, thereby requiring training data reflecting the desired trade-off to adapt the methods described here. We leave these application-centered issues for investigation in future work.

The current approach is primarily suitable for offline use, for example in report generation where there are no real-time interaction demands. In future work, we also plan to investigate ways that self-monitoring might be implemented more efficiently as a combined process, rather than running independent parsers as a post-process following realization.

Acknowledgments

We thank Mark Johnson, Micha Elsner, the OSU Clippers Group and the anonymous reviewers for helpful comments and discussion. This work was supported in part by NSF grants IIS-1143635 and IIS-1319318.

References

S. Abney. 1996. Statistical methods and linguistics. In Judith Klavans and Philip Resnik, editors, *The balancing act: Combining symbolic and statistical approaches to language*, pages 1–26. MIT Press, Cambridge, MA.

Anja Belz, Mike White, Josef van Genabith, Deirdre Hogan, and Amanda Stent. 2010. Finding common ground: Towards a surface realisation shared task. In *Proceedings of INLG-10, Generation Challenges*, pages 267–272.

Anja Belz, Michael White, Dominic Espinosa, Eric Kow, Deirdre Hogan, and Amanda Stent. 2011. The first surface realisation shared task: Overview and evaluation results. In *Proceedings of the Generation Challenges Session at the 13th European Workshop on Natural Language Generation*, pages 217–226, Nancy, France, September. Association for Computational Linguistics.

Stephen Boxwell and Michael White. 2008. Projecting Propbank roles onto the CCGbank. In *Proc. LREC-08*.

F. Chantree, B. Nuseibeh, A. De Roeck, and A. Willis. 2006. Identifying nocuous ambiguities in natural language requirements. In *Requirements Engineering, 14th IEEE International Conference*, pages 59–68. IEEE.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of ACL*, pages 173–180, Ann Arbor, Michigan. Association for Computational Linguistics.

Stephen Clark and James R. Curran. 2007. Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics*, 33(4):493–552.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proc. EMNLP-02*.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*.

Dominic Espinosa, Michael White, and Dennis Mehay. 2008. Hypertagging: Supertagging for surface realization with CCG. In *Proceedings of ACL-08: HLT*, pages 183–191, Columbus, Ohio, June. Association for Computational Linguistics.

Timothy A. D. Fowler and Gerald Penn. 2010. Accurate context-free parsing with Combinatory Categorial Grammar. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 335–344, Uppsala, Sweden, July. Association for Computational Linguistics.

Edward Gibson. 2000. Dependency locality theory: A distance-based theory of linguistic complexity. In Alec Marantz, Yasushi Miyashita, and Wayne O'Neil, editors, *Image, Language, brain: Papers from the First Mind Articulation Project Symposium*. MIT Press, Cambridge, MA.

Daniel Gildea and David Temperley. 2010. Do grammars minimize dependency length? *Cognitive Science*, 34(2):286–310.

Julia Hockenmaier and Mark Steedman. 2002. Generative models for statistical parsing with Combinatory Categorial Grammar. In *Proc. ACL-02*.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

T. Florian Jaeger. 2010. Redundancy and reduction: Speakers manage information density. *Cognitive Psychology*, 61(1):23–62, August.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proc. KDD*.

I.H. Khan, K. Van Deemter, and G. Ritchie. 2008. Generation of referring expressions: Managing structural ambiguities. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 433–440. Association for Computational Linguistics.

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

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain, July. Association for Computational Linguistics.

Willem J. M. Levelt. 1989. *Speaking: From Intention to Articulation*. MIT Press.

Günter Neumann and Gertjan van Noord. 1992. Self-monitoring with reversible grammars. In *Proceedings of the 14th conference on Computational linguistics - Volume 2*, COLING '92, pages 700–706, Stroudsburg, PA, USA. Association for Computational Linguistics.

Martha Palmer, Dan Gildea, and Paul Kingsbury. 2005. The proposition bank: A corpus annotated with semantic roles. *Computational Linguistics*, 31(1).

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. ACL-02*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of COLING-ACL*.

Rajakrishnan Rajkumar and Michael White. 2010. Designing agreement features for realization ranking. In *Proc. Coling 2010: Posters*, pages 1032–1040, Beijing, China, August.

Rajakrishnan Rajkumar and Michael White. 2011. Linguistically motivated complementizer choice in surface realization. In *Proceedings of the UCNLG+Eval: Language Generation and Evaluation Workshop*, pages 39–44, Edinburgh, Scotland, July. Association for Computational Linguistics.

Rajakrishnan Rajkumar, Michael White, and Dominic Espinosa. 2009. Exploiting named entity classes in CCG surface realization. In *Proc. NAACL HLT 2009 Short Papers*.

Ehud Reiter and Robert Dale. 2000. *Building natural generation systems*. Studies in Natural Language Processing. Cambridge University Press.

Ehud Reiter. 2010. Natural language generation. In Alexander Clark, Chris Fox, and Shalom Lappin, editors, *The Handbook of Computational Linguistics and Natural Language Processing (Blackwell Handbooks in Linguistics)*, Blackwell Handbooks in Linguistics, chapter 20. Wiley-Blackwell, 1 edition.

Mark Steedman. 2000. *The syntactic process*. MIT Press, Cambridge, MA, USA.

David Temperley. 2007. Minimization of dependency length in written English. *Cognition*, 105(2):300–333.

K. Van Deemter. 2004. Towards a probabilistic version of bidirectional OT syntax and semantics. *Journal of Semantics*, 21(3):251–280.

Michael White and Rajakrishnan Rajkumar. 2008. A more precise analysis of punctuation for broad-coverage surface realization with CCG. In *Coling 2008: Proceedings of the workshop on Grammar Engineering Across Frameworks*, pages 17–24.

Michael White and Rajakrishnan Rajkumar. 2009. Perceptron reranking for CCG realization. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 410–419, Singapore, August. Association for Computational Linguistics.

Michael White and Rajakrishnan Rajkumar. 2012. Minimal dependency length in realization ranking. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 244–255, Jeju Island, Korea, July. Association for Computational Linguistics.

Michael White. 2006. Efficient Realization of Coordinate Structures in Combinatory Categorial Grammar. *Research on Language & Computation*, 4(1):39–75.

Surface Realisation from Knowledge-Bases

Bikash Gyawali
Université de Lorraine, LORIA
Villers-lès-Nancy, F-54600, France
bikash.gyawali@loria.fr

Claire Gardent
CNRS, LORIA, UMR 7503
Vandoeuvre-lès-Nancy, F-54500, France
claire.gardent@loria.fr

Abstract

We present a simple, data-driven approach to generation from knowledge bases (KB). A key feature of this approach is that grammar induction is driven by the extended domain of locality principle of TAG (Tree Adjoining Grammar); and that it takes into account both syntactic and semantic information. The resulting extracted TAG includes a unification based semantics and can be used by an existing surface realiser to generate sentences from KB data. Experimental evaluation on the KBGen data shows that our model outperforms a data-driven generate-and-rank approach based on an automatically induced probabilistic grammar; and is comparable with a handcrafted symbolic approach.

1 Introduction

In this paper we present a grammar based approach for generating from knowledge bases (KB) which is linguistically principled and conceptually simple. A key feature of this approach is that grammar induction is driven by the extended domain of locality principle of TAG (Tree Adjoining Grammar) and takes into account both syntactic and semantic information. The resulting extracted TAGs include a unification based semantics and can be used by an existing surface realiser to generate sentences from KB data.

To evaluate our approach, we use the benchmark provided by the KBGen challenge (Banik et al., 2012; Banik et al., 2013), a challenge designed to evaluate generation from knowledge bases; where the input is a KB subset; and where the expected output is a complex sentence conveying the meaning represented by the input. When compared with two other systems having taken part in the KBGen challenge, our system outperforms a data-driven, generate-and-rank approach based on an automatically induced probabilistic grammar; and produces results comparable to those obtained by a symbolic, rule based approach. Most importantly, we obtain these results using a general purpose approach that we believe is simpler and more transparent than current state of the art surface realisation systems generating from KB or DB data.

2 Related Work

Our work is related to work on concept to text generation.

Earlier work on concept to text generation mainly focuses on generation from logical forms using rule-based methods. (Wang, 1980) uses hand-written rules to generate sentences from an extended predicate logic formalism; (Shieber et al., 1990) introduces a head-driven algorithm for generating from logical forms; (Kay, 1996) defines a chart based algorithm which enhances efficiency by minimising the number of semantically incomplete phrases being built; and (Shemtov, 1996) presents an extension of the chart based generation algorithm presented in (Kay, 1996) which supports the generation of multiple paraphrases from underspecified semantic input. In all these approaches, grammar and lexicon are developed manually and it is assumed that the lexicon associates semantic sub-formulae with natural language expressions. Our approach is similar to these approaches in that it assumes a grammar encoding a compositional semantics. It differs from them however in that, in our approach, grammar and lexicon are automatically acquired from the data.

With the development of the semantic web and the proliferation of knowledge bases, generation from knowledge bases has attracted increased interest and so called ontology verbalisers have been proposed which support the generation of text from (parts of) knowledge bases. One main

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 424–434,
Baltimore, Maryland, USA, June 23-25 2014.

strand of work maps each axiom in the knowledge base to a clause. Thus the OWL verbaliser integrated in the Protégé tool (Kaljurand and Fuchs, 2007) provides a verbalisation of every axiom present in the ontology under consideration and (Wilcock, 2003) describes an ontology verbaliser using XML-based generation. As discussed in (Power and Third, 2010), one important limitation of these approaches is that they assume a simple deterministic mapping between knowledge representation languages and some controlled natural language (CNL). Specifically, the assumption is that each atomic term (individual, class, property) maps to a word and each axiom maps to a sentence. As a result, the verbalisation of larger ontology parts can produce very unnatural text such as, *Every cat is an animal. Every dog is an animal. Every horse is an animal. Every rabbit is an animal.* More generally, the CNL based approaches to ontology verbalisation generate clauses (one per axiom) rather than complex sentences and thus cannot adequately handle the verbalisation of more complex input such as the KBGen data where the KB input often requires the generation of a complex sentence rather than a sequence of base clauses.

To generate more complex output from KB data, several alternative approaches have been proposed.

The MIAKT project (Bontcheva and Wilks., 2004) and the ONTOGENERATION project (Aguado et al., 1998) use symbolic NLG techniques to produce textual descriptions from some semantic information contained in a knowledge base. Both systems require some manual input (lexicons and domain schemas). More sophisticated NLG systems such as TAILOR (Paris, 1988), MIGRAINE (Mittal et al., 1994), and STOP (Reiter et al., 2003) offer tailored output based on user/patient models. While offering more flexibility and expressiveness, these systems are difficult to adapt by non-NLG experts because they require the user to understand the architecture of the NLG systems (Bontcheva and Wilks., 2004). Similarly, the NaturalOWL system (Galanis et al., 2009) has been proposed to generate fluent descriptions of museum exhibits from an OWL ontology. This approach however relies on extensive manual annotation of the input data.

The SWAT project has focused on producing descriptions of ontologies that are both coherent and efficient (Williams and Power, 2010). For instance, instead of the above output, the SWAT system would generate the sentence: *The following are kinds of animals: cats, dogs, horses and rabbits.* . In this approach too however, the verbaliser output is strongly constrained by a simple Definite Clause Grammar covering simple clauses and sentences verbalising aggregation patterns such as the above. More generally, the sentences generated by ontology verbalisers cover a limited set of linguistics constructions; the grammar used is manually defined; and the mapping between semantics and strings is assumed to be deterministic (e.g., a verb maps to a relation and a noun to a concept). In constrast, we propose an approach which can generate complex sentences from KB data; where the grammar is acquired from the data; and where no assumption is made about the mapping between semantics and NL expressions.

Recent work has focused on data-driven generation from frames, lambda terms and data base entries.

(DeVault et al., 2008) describes an approach for generating from the frames produced by a dialog system. They induce a probabilistic Tree Adjoining Grammar from a training set aligning frames and sentences using the grammar induction technique of (Chiang, 2000) and use a beam search that uses weighted features learned from the training data to rank alternative expansions at each step.

(Lu and Ng, 2011) focuses on generating natural language sentences from logical form (i.e., lambda terms) using a synchronous context-free grammar. They introduce a novel synchronous context free grammar formalism for generating from lambda terms; induce such a synchronous grammar using a generative model; and extract the best output sentence from the generated forest using a log linear model.

(Wong and Mooney, 2007; Lu et al., 2009) focuses on generating from variable-free tree-structured representations such as the CLANG formal language used in the ROBOCUP competition and the database entries collected by (Liang et al., 2009) for weather forecast generation and for the air travel domain (ATIS dataset) by (Dahl et al., 1994). (Wong and Mooney, 2007) uses synchronous grammars to transform a variable free tree structured meaning representation into sentences. (Lu et al., 2009) uses a Conditional Ran-

The function of a gated channel is to release particles from the endoplasmic reticulum

```
:TRIPLES (
(|Release-Of-Calcium646| |object| |Particle-In-Motion64582|)
(|Release-Of-Calcium646| |base| |Endoplasmic-Reticulum64603|)
(|Gated-Channel64605|  |has-function||Release-Of-Calcium646|)
(|Release-Of-Calcium646|  |agent| |Gated-Channel64605|))
:INSTANCE-TYPES
(|Particle-In-Motion64582| |instance-of| |Particle-In-Motion|)
(|Endoplasmic-Reticulum64603| |instance-of| |Endoplasmic-Reticulum|)
(|Gated-Channel64605| |instance-of| |Gated-Channel|)
 |Release-Of-Calcium646| |instance-of| |Release-Of-Calcium|))
:ROOT-TYPES (
(|Release-Of-Calcium646| |instance-of| |Event|)
(|Particle-In-Motion64582| |instance-of| |Entity|)
(|Endoplasmic-Reticulum64603| |instance-of| |Entity|)
(|Gated-Channel64605| |instance-of| |Entity|)))
```

Figure 1: Example KBGEN Scenario

dom Field to generate from the same meaning representations.

Finally, more recent papers propose approaches which perform both surface realisation and content selection. (Angeli et al., 2010) proposes a log linear model which decomposes into a sequence of discriminative local decisions. The first classifier determines which records to mention; the second, which fields of these records to select; and the third, which words to use to verbalise the selected fields. (Kim and Mooney, 2010) uses a generative model for content selection and verbalises the selected input using WASP^{-1}, an existing generator. Finally, (Konstas and Lapata, 2012b; Konstas and Lapata, 2012a) develop a joint optimisation approach for content selection and surface realisation using a generic, domain independent probabilistic grammar which captures the structure of the database and the mapping from fields to strings. They intersect the grammar with a language model to improve fluency; use a weighted hypergraph to pack the derivations; and find the best derivation tree using Viterbi algorithm.

Our approach differs from the approaches which assume variable free tree structured representations (Wong and Mooney, 2007; Lu et al., 2009) and data-based entries (Kim and Mooney, 2010; Konstas and Lapata, 2012b; Konstas and Lapata, 2012a) in that it handles graph-based, KB input and assumes a compositional semantics. It is closest to (DeVault et al., 2008) and (Lu and Ng, 2011) who extract a grammar encoding syntax and semantics from frames and lambda terms respectively. It differs from the former however in that it enforces a tighter syntax/semantics integration by requiring that the elementary trees of our extracted grammar encode the appropriate linking information. While (DeVault et al., 2008) extracts a TAG grammar associating each elementary tree with a semantics, we additionnally require that these trees encode the appropriate linking between syntactic and semantic arguments thereby restricting the space of possible tree combinations and drastically reducing the search space. Although conceptually related to (Lu and Ng, 2011), our approach extracts a unification based grammar rather than one with lambda terms. The extraction process and the generation algorithms are also fundamentally different. We use a simple mainly symbolic approach whereas they use a generative approach for grammar induction and a discriminative approach for sentence generation.

3 The KBGen Task

The KBGen task was introduced as a new shared task at Generation Challenges 2013 (Banik et al., 2013)[1] and aimed to compare different generation systems on KB data. Specifically, the task is to verbalise a subset of a knowledge base. For instance, the KB input shown in Figure 1 can be verbalised as:

(1) The function of a gated channel is to release particles from the endoplasmic reticulum

The KB subsets forming the KBGen input data were pre-selected from the AURA biology knowledge base (Gunning et al., 2010), a knowledge base about biology which was manually encoded by biology teachers and encodes knowledge about events, entities, properties and relations where relations include event-to-entity, event-to-event,

[1] http://www.kbgen.org

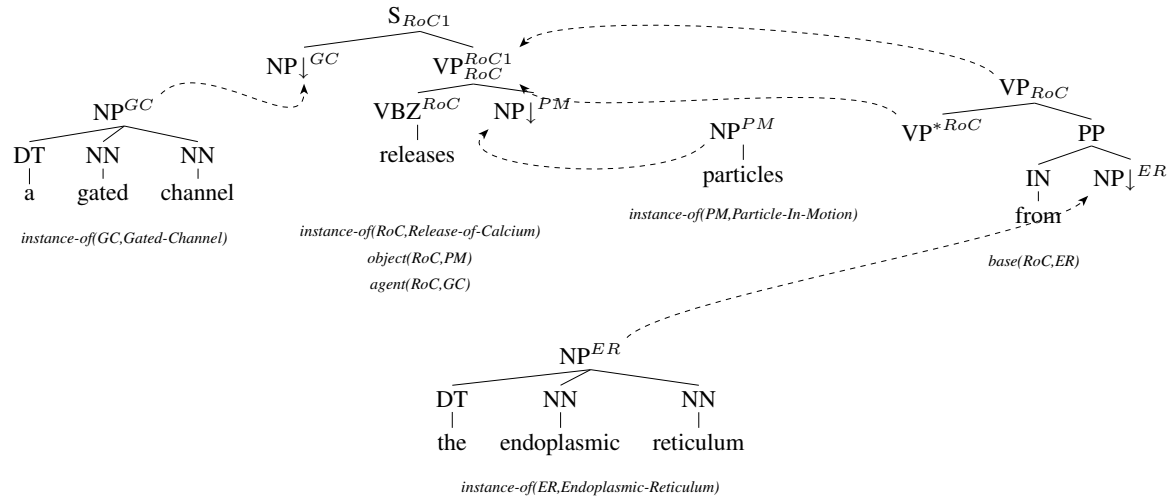


Figure 2: Example FB-LTAG with Unification-Based Semantics. Dotted lines indicate substitution and adjunction operations between trees. The variables decorating the tree nodes (e.g., GC) abbreviate feature structures of the form $[idx : V]$ where V is a unification variable shared with the semantics.

event-to-property and entity-to-property relations. AURA uses a frame-based knowledge representation and reasoning system called Knowledge Machine (Clark and Porter, 1997) which was translated into first-order logic with equality and from there, into multiple different formats including SILK (Grosof, 2012) and OWL2 (Motik et al., 2009). It is available for download in various formats including OWL[2].

4 Generating from the KBGen Knowledge-Base

To generate from the KBGen data, we induce a Feature-Based Lexicalised Tree Adjoining Grammar (FB-LTAG, (Vijay-Shanker and Joshi, 1988)) augmented with a unification-based semantics (Gardent and Kallmeyer, 2003) from the training data. We then use this grammar and an existing surface realiser to generate from the test data.

4.1 Feature-Based Lexicalised Tree Adjoining Grammar

Figure 2 shows an example FB-LTAG augmented with a unification-based semantics.

Briefly, an FB-LTAG consists of a set of elementary trees which can be either initial or auxiliary. Initial trees are trees whose leaves are labeled with substitution nodes (marked with a down-arrow) or terminal categories. Auxiliary trees are distinguished by a foot node (marked with a star) whose category must be the same as that of the root node. In addition, in an FB-LTAG, each elementary tree is anchored by a lexical item (lexicalisation) and the nodes in the elementary trees are decorated with two feature structures called *top* and *bottom* which are unified during derivation. Two tree-composition operations are used to combine trees namely, substitution and adjunction. While substitution inserts a tree in a substitution node of another tree, adjunction inserts an auxiliary tree into a tree. In terms of unifications, substitution unifies the top feature structure of the substitution node with the top feature structure of the root of the tree being substituted in. Adjunction unifies the top feature structure of the root of the tree being adjoined with the top feature structure of the node being adjoined to; and the bottom feature structure of the foot node of the auxiliary tree being adjoined with the bottom feature structure of the node being adjoined to.

In an FB-LTAG augmented with a unification-based semantics, each tree is associated with a semantics i.e., a set of literals whose arguments may be constants or unification variables. The semantics of a derived tree is the union of the semantics of the tree contributing to its derivation modulo unification. Importantly, semantic variables are shared with syntactic variables (i.e., variables occurring in the feature structures decorating the tree nodes) so that when trees are combined, the appropriate syntax/semantics linking is enforced. For instance given the semantics:

[2] http://www.ai.sri.com/halo/halobook2010/exported-kb/biokb.html

instance-of(RoC,Release-Of-Calcium),
object(RoC,PM),agent(RoC,GC),base(RoC,ER),
instance-of(ER,Endoplasmic-Reticulum),
instance-of(GC,Gated-Channel),
instance-of(PM,Particle-In-Motion)

the grammar will generate *A gated channel releases particles from the endoplasmic reticulum* but not e.g., *Particles releases a gated channel from the endoplasmic reticulum*.

4.2 Grammar Extraction

We extract our FB-LTAG with unification semantics from the KBGen training data in two main steps. First, we align the KB data with the input string. Second, we induce a Tree Adjoining Grammar augmented with a unification-based semantics from the aligned data.

4.2.1 Alignment

Given a Sentence/Input pair (S, I) provided by the KBGen Challenge, the alignment procedure associates each entity and event variable in I to a substring in S. To do this, we use the entity and the event lexicon provided by the KBGen organiser. The event lexicon maps event types to verbs, their inflected forms and nominalizations while the entity lexicon maps entity types to a noun and its plural form. For instance, the lexicon entries for the event and entity types shown in Figure 1 are as shown in Figure 3.

For each entity and each event variable V in I, we retrieve the corresponding type (e.g., `Particle-In-Motion` for `Particle-In-Motion64582`); search the KBGen lexicon for the corresponding phrases (e.g., *molecule in motion,molecules in motion*); and associate V with the phrase in S which matches one of these phrases. Figure 3 shows an example lexicon and the resulting alignment obtained for the scenario shown in Figure 1. Note that there is not always an exact match between the phrase associated in the KBGen lexicon with a type and the phrase occurring in the training sentence. To account for this, we use some additional similarity based heuristics to identify the phrase in the input string that is most likely to be associated with a variable lacking an exact match in the input string. E.g., for entity variables (e.g., `Particle-In-Motion64582`), we search the input string for nouns (e.g., particles) whose overlap with the variable type (e.g., Particle-In-Motion) is not empty.

4.2.2 Inducing a based FB-LTAG from the aligned data

To extract a Feature-Based Lexicalised Tree Adjoining Grammar (FB-LTAG) from the KBGen data, we parse the sentences of the training corpus; project the entity and event variables to the syntactic projection of the strings they are aligned with; and extract the elementary trees of the resulting FB-LTAG from the parse tree using semantic information. Figure 4 shows the trees extracted from the scenario given in Figure 1.

To associate each training example sentence with a syntactic parse, we use the Stanford parser. After alignment, the entity and event variables occurring in the input semantics are associated with substrings of the yield of the syntactic parse tree. We project these variables up the syntactic tree to reflect headedness. A variable aligned with a noun is projected to the NP level or to the immediately dominating PP if it occurs in the subtree dominated by the leftmost daughter of that PP. A variable aligned with a verb is projected to the first S node immediately dominating that verb or, in the case of a predicative sentence, to the root of that sentence[3].

Once entity and event variables have been projected up the parse trees, we extract elementary FB-LTAG trees and their semantics from the input scenario as follows.

First, the subtrees whose root node is indexed with an entity variable are extracted. This results in a set of NP and PP trees anchored with entity names and associated with the predication true of the indexing variable.

Second, the subtrees capturing relations between variables are extracted. To perform this extraction, each input variable X is associated with a set of dependent variables i.e., the set of variables Y such that X is related to Y $(R(X, Y))$. The minimal tree containing all and only the dependent variables $D(X)$ of a variable X is then extracted and associated with the set of literals Φ such that $\Phi = \{R(Y, Z) \mid (Y = X \wedge Z \in D(X)) \vee (Y, Z \in D(X))\}$. This procedure extracts the subtrees relating the argument variables of a semantic functors such as an event or a role e.g., a tree describing a verb and its arguments as shown in the top

[3]Initially, we used the head information provided by the Stanford parser. In practice however, we found that the heuristics we defined to project semantic variables to the corresponding syntactic projection were more accurate and better supported our grammar extraction process.

`Particle-In-Motion`	molecule in motion,molecules in motion
`Endoplasmic-Reticulum`	endoplasmic reticulum,endoplasmic reticulum
`Gated-Channel`	gated Channel,gated Channels
`Release-Of-Calcium`	releases,release,released,release

The function of a (gated channel, `Gated-Channel64605`) is to (release, `Release-Of-Calcium646`) (particles, `Particle-In-Motion64582`) from the (endoplasmic reticulum, `Endoplasmic-Reticulum64603`)

Figure 3: Example Entries from the KBGen Lexicon and example alignment

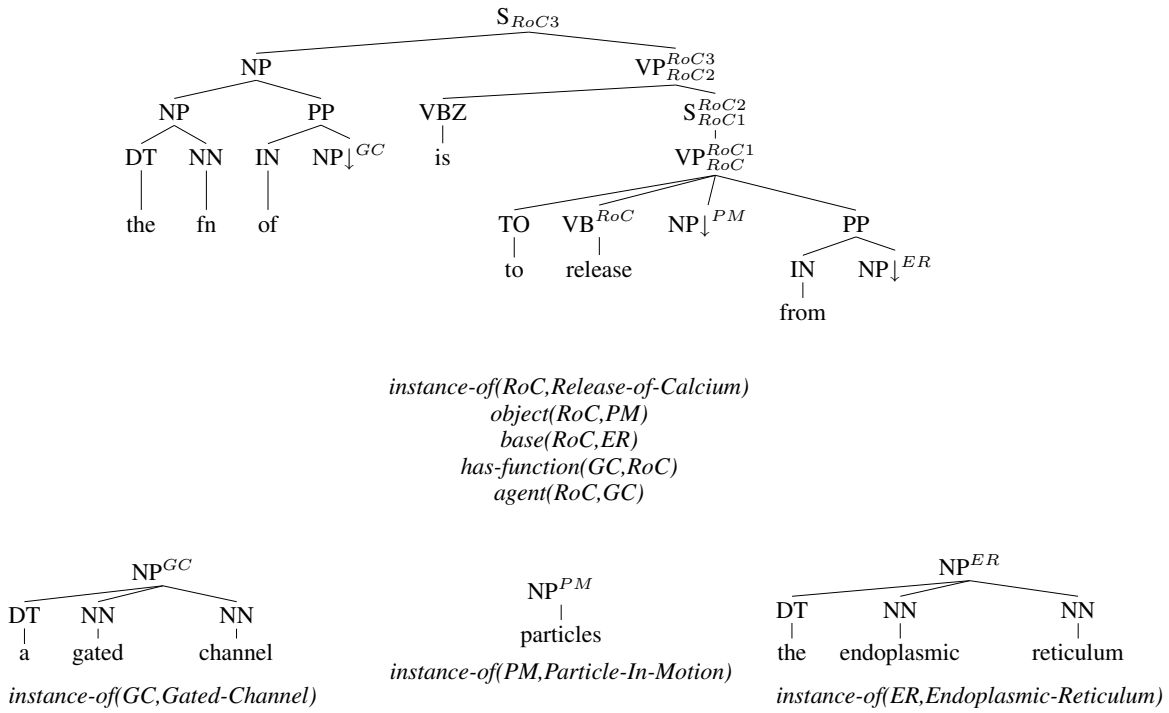


Figure 4: Extracted Grammar for "*The function of a gated channel is to release particles from the endoplasmic reticulum*". Variable names have been abbreviated and the KBGen tuple notation converted to terms so as to fit the input format expected by our surface realiser.

part of Figure 4. Note that such a tree may capture a verb occurring in a relative or a subordinate clause (together with its arguments) thus allowing for complex sentences including a relative or relating a main and a subordinate clause.

The resulting grammar extracted from the parse trees (cf. e.g., Figure 4) is a Feature-Based Tree Adjoining Grammar with a Unification-based compositional semantics as described in (Gardent and Kallmeyer, 2003). In particular, our grammars differs from the traditional probabilistic Tree Adjoining Grammar extracted as described in e.g., (Chiang, 2000) in that they encode both syntax and semantics rather than just syntax. They also differ from the semantic FB-TAG extracted by (DeVault et al., 2008) in that (i) they encode the linking between syntactic and semantic arguments; (ii) they allow for elementary trees spanning discontiguous strings (e.g., *The function of X is to release Y*); and (iii) they enforce the semantic principle underlying TAG namely that an elementary tree containing a syntactic functor also contains its syntactic arguments.

4.3 Generation

To generate with the grammar extracted from the KBGen data, we use the GenI surface realiser (Gardent et al., 2007). Briefly, given an input semantics and a FB-LTAG with a unification based semantics, GenI selects all grammar entries whose semantics subsumes the input semantics; combines these entries using the FB-LTAG combination operations (i.e., adjunction and substitution); and outputs the yield of all derived trees which are syntactically complete and whose semantics is the input semantics. To rank the generator output, we train a language model on the GeniA corpus [4], a corpus of 2000 MEDLINE asbtracts about biology containing more than 400000 words (Kim et al., 2003) and use this model to rank the generated sentences by decreasing probability.

Thus for instance, given the input semantics shown in Figure 1 and the grammar depicted in Figure 4, the surface realiser will select all of these trees; combine them using FB-LTAG substitution operation; and output as generated sentence the yield of the resulting derived tree namely the sentence *The function of a gated channel is to release particles from the endoplasmic reticulum*.

However, this procedure only works if the entries necessary to generate from the given input are present in the grammar. To handle new, unseen input, we proceed in two ways. First, we try to guess a grammar entry from the shape of the input and the existing grammar. Second, we expand the grammar by decomposing the extracted trees into simpler ones.

4.4 Guessing new grammar entries.

Given the limited size of the training data, it is often the case that input from the test data will have no matching grammar unit. To handle such previously unseen input, we start by partitioning the input semantics into sub-semantics corresponding to events, entities and role.

For each entity variable X of type $Type$, we create a default NP tree whose semantics is a literal of the form *instance-of(X,Type)*.

For event variables, we search the lexicon for an entry with a matching or similar semantics i.e., an entry with the same number and same type of literals (literals with same arity and with identical relations). When one is found, a grammar entry is constructed for the unseen event variable by substituting the event type of the matching entry with the type of the event variable. For instance, given the input semantics *instance-of(C,Carry), object(C,X), base(C,Y), has-function(Z,C), agent(C,Z)*, this procedure will create a grammar entry identical to that shown at the top of Figure 4 except that the event type *Release-of-Calcium* is changed to *Carry* and the terminal *release* to the word form associated in the KBGen lexicon with this concept, namely to the verb *carry*.

4.5 Expanding the Grammar

While the extracted grammar nicely captures predicate/argument dependencies, it is very specific to the items seen in the training data. To reduce overfitting, we generalise the extracted grammar by extracting from each event tree, subtrees that capture structures with fewer arguments and optional modifiers.

For each event tree τ extracted from the training data which contains a subject-verb-object subtree τ', we add τ' to the grammar and associate it with the semantics of τ minus the relations associated with the arguments that have been removed. For instance, given the extracted tree for the sentence *"Aquaporin facilitates the movement of water molecules through hydrophilic channels."*, this

[4] http://www.nactem.ac.uk/genia/

procedure will construct a new grammar tree corresponding to the subphrase *"Aquaporin facilitates the movement of water molecules"*.

We also construct both simpler event trees and optional modifiers trees by extracting from event trees, PP trees which are associated with a relational semantics. For instance, given the tree shown in Figure 4, the PP tree associated with the relation *base(RoC,ET)* is removed thus creating two new trees as illustrated in Figure 5: an S tree corresponding to the sentence *The function of a gated channel is to release particles* and an auxiliary PP tree corresponding to the phrase *from the endoplasmic reticulum*. Similarly in the above example, a PP tree corresponding to the phrase *"through hydrophilic channels."* will be extracted.

As with the base grammar, missing grammar entries are guessed from the expanded grammar. However we do this only in cases where a correct grammar entry cannot be guessed from the base grammar.

5 Experimental Setup

We evaluate our approach on the KBGen data and compare it with the KBGen reference and two other systems having taken part to the KBGen challenge.

5.1 Training and test data.

Following a practice introduced by (Angeli et al., 2010), we use the term *scenario* to denote a KB subset paired with a sentence. The KBGen benchmark contains 207 scenarii for training and 72 for testing. Each KB subset consists of a set of triples and each scenario contains on average 16 triples and 17 words.

5.2 Systems

We evaluate three configurations of our approach on the KBGen test data: one without grammar expansion (BASE); a second with a manual grammar expansion MANEXP; and a third one with automated grammar expansion AUTEXP. We compare the results obtained with those obtained by two other systems participating in the KBGen challenge, namely the UDEL system, a symbolic rule based system developed by a group of students at the University of Delaware; and the IMS system, a statistical system using a probabilistic grammar induced from the training data.

5.3 Metrics.

We evaluate system output automatically, using the BLEU-4 modified precision score (Papineni et al., 2002) with the human written sentences as reference. We also report results from a human based evaluation. In this evaluation, participants were asked to rate sentences along three dimensions: **fluency** (Is the text easy to read?), **grammaticality** and meaning similarity or **adequacy** (Does the meaning conveyed by the generated sentence correspond to the meaning conveyed by the reference sentence?). The evaluation was done on line using the LG-Eval toolkit (Kow and Belz, 2012), subjects used a sliding scale from -50 to +50 and a Latin Square Experimental Design was used to ensure that each evaluator sees the same number of outputs from each system and for each test set item. 12 subjects participated in the evaluation and 3 judgments were collected for each output.

6 Results and Discussion

System	All	Covered	Coverage	# Trees
IMS	0.12	0.12	100%	
UDEL	0.32	0.32	100%	
Base	0.04	0.39	30.5%	371
ManExp	0.28	0.34	83 %	412
AutExp	0.29	0.29	100%	477

Figure 6: BLEU scores and Grammar Size (Number of Elementary TAG trees

Table 6 summarises the results of the automatic evaluation and shows the size (number of elementary TAG trees) of the grammars extracted from the KBGen data.

The average BLEU score is given with respect to all input (All) and to those inputs for which the systems generate at least one sentence (Covered). While both the IMS and the UDEL system have full coverage, our BASE system strongly undergenerates failing to account for 69.5% of the test data. However, because the extracted grammar is linguistically principled and relatively compact, it is possible to manually edit it. Indeed, the MANEXP results show that, by adding 41 trees to the grammar, coverage can be increased by 52.5 points reaching a coverage of 83%. Finally, the AUTEXP results demonstrate that the automated expansion mechanism permits achieving full coverage while keeping a relative small grammar (477 trees).

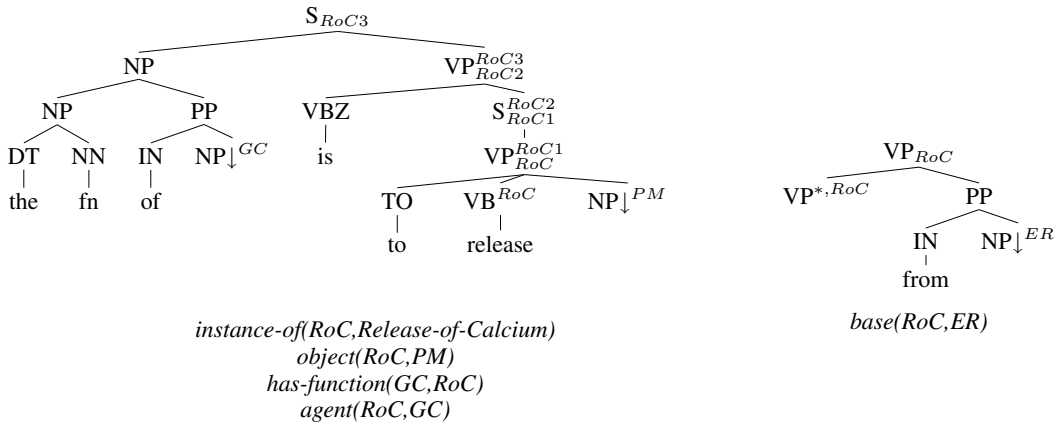


Figure 5: Trees Added by the Expansion Process

	Fluency		Grammaticality		Meaning Similarity	
System	Mean	Homogeneous Subsets	Mean	Homogeneous Subsets	Mean	Homogeneous Subsets
UDEL	4.36	A	4.48	A	3.69	A
AutExp	3.45	B	3.55	B	3.65	A
IMS	1.91	C	2.05	C	1.31	B

Figure 7: Human Evaluation Results on a scale of 0 to 5. Homogeneous subsets are determined using Tukey's Post Hoc Test with $p < 0.05$

In terms of BLEU score, the best version of our system (AUTEXP) outperforms the probabilistic approach of IMS by a large margin (+0.17) and produces results similar to the fully handcrafted UDEL system (-0.03).

In sum, our approach permits obtaining BLEU scores and a coverage which are similar to that obtained by a hand crafted system and outperforms a probabilistic approach. One key feature of our approach is that the grammar extracted from the training data is linguistically principled in that it obeys the extended locality principle of Tree Adjoining Grammars. As a result, the extracted grammar is compact and can be manually modified to fit the need of an application as shown by the good results obtained when using the MANEXP configuration.

We now turn to the results of the human evaluation. Table 7 summarises the results whereby systems are grouped by letters when there is no significant difference between them (significance level: $p < 0.05$). We used ANOVAs and post-hoc Tukey tests to test for significance. The differences between systems are statistically significant throughout except for meaning similarity (adequacy) where UDEL and our system are on the same level. Across the metrics, our system consistently ranks second behind the symbolic, UDEL system and before the statistical IMS one thus confirming the ranking based on BLEU.

7 Conclusion

In Tree Adjoining Grammar, the *extended domain of locality principle* ensures that TAG trees group together in a single structure a syntactic predicate and its arguments. Moreover, the *semantic principle* requires that each elementary tree captures a single semantic unit. Together these two principles ensure that TAG elementary trees capture basic semantic units and their dependencies. In this paper, we presented a grammar extraction approach which ensures that extracted grammars comply with these two basic TAG principles. Using the KBGen benchmark, we then showed that the resulting induced FB-LTAG compares favorably with competing symbolic and statistical approaches when used to generate from knowledge base data.

In the current version of the generator, the output is ranked using a simple language model trained on the GENIA corpus. We observed that this often fails to return the best output in terms of BLEU score, fluency, grammaticality and/or meaning. In the future, we plan to remedy this using a ranking approach such as proposed in (Velldal and Oepen, 2006; White and Rajkumar, 2009).

References

G. Aguado, A. Bañón, J. Bateman, S. Bernardos, M. Fernández, A. Gómez-Pérez, E. Nieto, A. Olalla, R. Plaza, and A. Sánchez. 1998. Ontogeneration: Reusing domain and linguistic ontologies for spanish text generation. In *Workshop on Applications of Ontologies and Problem Solving Methods, ECAI*, volume 98.

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 502–512. Association for Computational Linguistics.

Eva Banik, Claire Gardent, Donia Scott, Nikhil Dinesh, and Fennie Liang. 2012. Kbgen: Text generation from knowledge bases as a new shared task. In *Proceedings of the seventh International Natural Language Generation Conference*, pages 141–145. Association for Computational Linguistics.

Eva Banik, Claire Gardent, Eric Kow, et al. 2013. The kbgen challenge. In *Proceedings of the 14th European Workshop on Natural Language Generation (ENLG)*, pages 94–97.

K. Bontcheva and Y. Wilks. 2004. Automatic report generation from ontologies: the miakt approach. In *Ninth International Conference on Applications of Natural Language to Information Systems (NLDB'2004)*. Lecture Notes in Computer Science 3136, Springer, Manchester, UK.

David Chiang. 2000. Statistical parsing with an automatically-extracted tree adjoining grammar. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 456–463. Association for Computational Linguistics.

Peter Clark and Bruce Porter. 1997. Building concept representations from reusable components. In *AAAI/IAAI*, pages 369–376. Citeseer.

Deborah A Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the atis task: The atis-3 corpus. In *Proceedings of the workshop on Human Language Technology*, pages 43–48. Association for Computational Linguistics.

David DeVault, David Traum, and Ron Artstein. 2008. Making grammar-based generation easier to deploy in dialogue systems. In *Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue*, pages 198–207. Association for Computational Linguistics.

D. Galanis, G. Karakatsiotis, G. Lampouras, and I. Androutsopoulos. 2009. An open-source natural language generator for owl ontologies and its use in protégé and second life. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics: Demonstrations Session*, pages 17–20. Association for Computational Linguistics.

Claire Gardent and Laura Kallmeyer. 2003. Semantic construction in feature-based tag. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1*, pages 123–130. Association for Computational Linguistics.

Claire Gardent, Eric Kow, et al. 2007. A symbolic approach to near-deterministic surface realisation using tree adjoining grammar. In *ACL*, volume 7, pages 328–335.

B. Grosof. 2012. The silk project: Semantic inferencing on large knowledge. Technical report, SRI. `http://silk.semwebcentral.org/`.

D. Gunning, V. K. Chaudhri, P. Clark, K. Barker, Shaw-Yi Chaw, M. Greaves, B. Grosof, A. Leung, D. McDonald, S. Mishra, J. Pacheco, B. Porter, A. Spaulding, D. Tecuci, and J. Tien. 2010. Project halo update - progress toward digital aristotle. *AI Magazine*, Fall:33–58.

K. Kaljurand and N.E. Fuchs. 2007. Verbalizing owl in attempto controlled english. *Proceedings of OWLED07*.

Martin Kay. 1996. Chart generation. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 200–204. Association for Computational Linguistics.

Joohyun Kim and Raymond J Mooney. 2010. Generative alignment and semantic parsing for learning from ambiguous supervision. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 543–551. Association for Computational Linguistics.

J-D Kim, Tomoko Ohta, Yuka Tateisi, and Junichi Tsujii. 2003. Genia corpusa semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl 1):i180–i182.

Ioannis Konstas and Mirella Lapata. 2012a. Concept-to-text generation via discriminative reranking. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 369–378. Association for Computational Linguistics.

Ioannis Konstas and Mirella Lapata. 2012b. Unsupervised concept-to-text generation with hypergraphs. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 752–761. Association for Computational Linguistics.

Eric Kow and Anja Belz. 2012. Lg-eval: A toolkit for creating online language evaluation experiments. In *LREC*, pages 4033–4037.

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 91–99. Association for Computational Linguistics.

Wei Lu and Hwee Tou Ng. 2011. A probabilistic forest-to-string model for language generation from typed lambda calculus expressions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1611–1622. Association for Computational Linguistics.

Wei Lu, Hwee Tou Ng, and Wee Sun Lee. 2009. Natural language generation with tree conditional random fields. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 400–409. Association for Computational Linguistics.

VO Mittal, G. Carenini, and JD Moore. 1994. Generating patient specific explanations in migraine. In *Proceedings of the eighteenth annual symposium on computer applications in medical care. McGraw-Hill Inc*.

Boris Motik, Peter F Patel-Schneider, Bijan Parsia, Conrad Bock, Achille Fokoue, Peter Haase, Rinke Hoekstra, Ian Horrocks, Alan Ruttenberg, Uli Sattler, et al. 2009. Owl 2 web ontology language: Structural specification and functional-style syntax. *W3C recommendation*, 27:17.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

C.L. Paris. 1988. Tailoring object descriptions to a user's level of expertise. *Computational Linguistics*, 14(3):64–78.

R. Power and A. Third. 2010. Expressing owl axioms by english sentences: dubious in theory, feasible in practice. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1006–1013. Association for Computational Linguistics.

E. Reiter, R. Robertson, and L.M. Osman. 2003. Lessons from a failure: Generating tailored smoking cessation letters. *Artificial Intelligence*, 144(1):41–58.

Hadar Shemtov. 1996. Generation of paraphrases from ambiguous logical forms. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*, pages 919–924. Association for Computational Linguistics.

Stuart M Shieber, Gertjan Van Noord, Fernando CN Pereira, and Robert C Moore. 1990. Semantic-head-driven generation. *Computational Linguistics*, 16(1):30–42.

Erik Velldal and Stephan Oepen. 2006. Statistical ranking in tactical generation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 517–525. Association for Computational Linguistics.

K. Vijay-Shanker and AK Joshi. 1988. Feature structures based tree adjoining grammars. In *Proceedings of the 12th International Conference on Computational Linguistics*, Budapest, Hungary.

Juen-tin Wang. 1980. On computational sentence generation from logical form. In *Proceedings of the 8th conference on Computational linguistics*, pages 405–411. Association for Computational Linguistics.

Michael White and Rajakrishnan Rajkumar. 2009. Perceptron reranking for ccg realization. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 410–419. Association for Computational Linguistics.

G. Wilcock. 2003. Talking owls: Towards an ontology verbalizer. *Human Language Technology for the Semantic Web and Web Services, ISWC*, 3:109–112.

Sandra Williams and Richard Power. 2010. Grouping axioms for more coherent ontology descriptions. In *Proceedings of the 6th International Natural Language Generation Conference (INLG 2010)*, pages 197–202, Dublin.

Yuk Wah Wong and Raymond J Mooney. 2007. Generation by inverting a semantic parser that uses statistical machine translation. In *HLT-NAACL*, pages 172–179.

Hybrid Simplification using Deep Semantics and Machine Translation

Shashi Narayan
Université de Lorraine, LORIA
Villers-lès-Nancy, F-54600, France
shashi.narayan@loria.fr

Claire Gardent
CNRS, LORIA, UMR 7503
Vandoeuvre-lès-Nancy, F-54500, France
claire.gardent@loria.fr

Abstract

We present a hybrid approach to sentence simplification which combines deep semantics and monolingual machine translation to derive simple sentences from complex ones. The approach differs from previous work in two main ways. First, it is semantic based in that it takes as input a deep semantic representation rather than e.g., a sentence or a parse tree. Second, it combines a simplification model for splitting and deletion with a monolingual translation model for phrase substitution and reordering. When compared against current state of the art methods, our model yields significantly simpler output that is both grammatical and meaning preserving.

1 Introduction

Sentence simplification maps a sentence to a simpler, more readable one approximating its content. Typically, a simplified sentence differs from a complex one in that it involves simpler, more usual and often shorter, words (e.g., *use* instead of *exploit*); simpler syntactic constructions (e.g., no relative clauses or apposition); and fewer modifiers (e.g., *He slept* vs. *He also slept*). In practice, simplification is thus often modeled using four main operations: *splitting* a complex sentence into several simpler sentences; *dropping* and *reordering* phrases or constituents; *substituting* words/phrases with simpler ones.

As has been argued in previous work, sentence simplification has many potential applications. It is useful as a preprocessing step for a variety of NLP systems such as parsers and machine translation systems (Chandrasekar et al., 1996), summarisation (Knight and Marcu, 2000), sentence fusion (Filippova and Strube, 2008) and semantic role labelling (Vickrey and Koller, 2008). It also has wide ranging potential societal application as a reading aid for people with aphasis (Carroll et al., 1999), for low literacy readers (Watanabe et al., 2009) and for non native speakers (Siddharthan, 2002).

There has been much work recently on developing computational frameworks for sentence simplification. Synchronous grammars have been used in combination with linear integer programming to generate and rank all possible rewrites of an input sentence (Dras, 1999; Woodsend and Lapata, 2011). Machine Translation systems have been adapted to translate complex sentences into simple ones (Zhu et al., 2010; Wubben et al., 2012; Coster and Kauchak, 2011). And handcrafted rules have been proposed to model the syntactic transformations involved in simplifications (Siddharthan et al., 2004; Siddharthan, 2011; Chandrasekar et al., 1996).

In this paper, we present a hybrid approach to sentence simplification which departs from this previous work in two main ways.

First, it combines a model encoding probabilities for splitting and deletion with a monolingual machine translation module which handles reordering and substitution. In this way, we exploit the ability of statistical machine translation (SMT) systems to capture phrasal/lexical substitution and reordering while relying on a dedicated probabilistic module to capture the splitting and deletion operations which are less well (deletion) or not at all (splitting) captured by SMT approaches.

Second, our approach is semantic based. While previous simplification approaches starts from either the input sentence or its parse tree, our model takes as input a deep semantic representation namely, the Discourse Representation Structure (DRS, (Kamp, 1981)) assigned by Boxer (Curran et al., 2007) to the input complex sentence. As we

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 435–445,
Baltimore, Maryland, USA, June 23-25 2014.

shall see in Section 4, this permits a linguistically principled account of the splitting operation in that semantically shared elements are taken to be the basis for splitting a complex sentence into several simpler ones; this facilitates completion (the re-creation of the shared element in the split sentences); and this provide a natural means to avoid deleting obligatory arguments.

When compared against current state of the art methods (Zhu et al., 2010; Woodsend and Lapata, 2011; Wubben et al., 2012), our model yields significantly simpler output that is both grammatical and meaning preserving.

2 Related Work

Earlier work on sentence simplification relied on handcrafted rules to capture syntactic simplification e.g., to split coordinated and subordinated sentences into several, simpler clauses or to model active/passive transformations (Siddharthan, 2002; Chandrasekar and Srinivas, 1997; Bott et al., 2012; Canning, 2002; Siddharthan, 2011; Siddharthan, 2010). While these handcrafted approaches can encode precise and linguistically well-informed syntactic transformation (using e.g., detailed morphological and syntactic information), they are limited in scope to purely syntactic rules and do not account for lexical simplifications and their interaction with the sentential context.

Using the parallel dataset formed by Simple English Wikipedia (SWKP)[1] and traditional English Wikipedia (EWKP)[2], more recent work has focused on developing machine learning approaches to sentence simplification.

Zhu et al. (2010) constructed a parallel corpus (PWKP) of 108,016/114,924 complex/simple sentences by aligning sentences from EWKP and SWKP and used the resulting bitext to train a simplification model inspired by *syntax-based* machine translation (Yamada and Knight, 2001). Their simplification model encodes the probabilities for four rewriting operations on the parse tree of an input sentences namely, substitution, reordering, splitting and deletion. It is combined with a language model to improve grammaticality and the decoder translates sentences into simpler ones by greedily selecting the output sentence with highest probability.

Using both the PWKP corpus developed by Zhu et al. (2010) and the edit history of Simple Wikipedia, Woodsend and Lapata (2011) learn a quasi synchronous grammar (Smith and Eisner, 2006) describing a loose alignment between parse trees of complex and of simple sentences. Following Dras (1999), they then generate all possible rewrites for a source tree and use integer linear programming to select the most appropriate simplification. They evaluate their model on the same dataset used by Zhu et al. (2010) namely, an aligned corpus of 100/131 EWKP/SWKP sentences and show that they achieve better BLEU score. They also conducted a human evaluation on 64 of the 100 test sentences and showed again a better performance in terms of simplicity, grammaticality and meaning preservation.

In (Wubben et al., 2012; Coster and Kauchak, 2011), simplification is viewed as a monolingual translation task where the complex sentence is the source and the simpler one is the target. To account for deletions, reordering and substitution, Coster and Kauchak (2011) trained a phrase based machine translation system on the PWKP corpus while modifying the word alignment output by GIZA++ in Moses to allow for null phrasal alignments. In this way, they allow for phrases to be deleted during translation. No human evaluation is provided but the approach is shown to result in statistically significant improvements over a traditional phrase based approach. Similarly, Wubben et al. (2012) use Moses and the PWKP data to train a phrase based machine translation system augmented with a post-hoc reranking procedure designed to rank the output based on their dissimilarity from the source. A human evaluation on 20 sentences randomly selected from the test data indicates that, in terms of fluency and adequacy, their system is judged to outperform both Zhu et al. (2010) and Woodsend and Lapata (2011) systems.

3 Simplification Framework

We start by motivating our approach and explaining how it relates to previous proposals w.r.t., the four main operations involved in simplification namely, splitting, deletion, substitution and reordering. We then introduce our framework.

[1] SWKP (`http://simple.wikipedia.org`) is a corpus of simple texts targeting "children and adults who are learning English Language" and whose authors are requested to "use easy words and short sentences".

[2] `http://en.wikipedia.org`

Sentence Splitting. Sentence splitting is arguably semantic based in that in many cases, splitting occurs when the same semantic entity participates in two distinct eventualities. For instance, in example (1) below, the split is on the noun *bricks* which is involved in two eventualities namely, *"being resistant to cold"* and *"enabling the construction of permanent buildings"*.

(1) **C.** Being more resistant to cold, bricks enabled the construction of permanent buildings.
S. Bricks were more resistant to cold. Bricks enabled the construction of permanent buildings.

While splitting opportunities have a clear counterpart in syntax (i.e., splitting often occurs whenever a relative, a subordinate or an appositive clause occurs in the complex sentence), completion i.e., the reconstruction of the shared element in the second simpler clause, is arguably semantically governed in that the reconstructed element corefers with its matching phrase in the first simpler clause. While our semantic based approach naturally accounts for this by copying the phrase corresponding to the shared entity in both phrases, syntax based approach such as Zhu et al. (2010) and Woodsend and Lapata (2011) will often fail to appropriately reconstruct the shared phrase and introduce agreement mismatches because the alignment or rules they learn are based on syntax alone. For instance, in example (2), Zhu et al. (2010) fails to copy the shared argument *"The judge"* to the second clause whereas Woodsend and Lapata (2011) learns a synchronous rule matching (VP and VP) to (VP. NP(It) VP) thereby failing to produce the correct subject pronoun (*"he"* or *"she"*) for the antecedent *"The judge"*.

(2) **C.** The judge ordered that Chapman should receive psychiatric treatment in prison and sentenced him to twenty years to life.
S_1. The judge ordered that Chapman should get psychiatric treatment. In prison and sentenced him to twenty years to life. (Zhu et al., 2010)
S_2. The judge ordered that Chapman should receive psychiatric treatment in prison. It sentenced him to twenty years to life. (Woodsend and Lapata, 2011)

Deletion. By handling deletion using a probabilistic model trained on semantic representations, we can avoid deleting obligatory arguments. Thus in our approach, semantic subformulae which are related to a predicate by a core thematic roles (e.g., *agent* and *patient*) are never considered for deletion. By contrast, syntax based approaches (Zhu et al., 2010; Woodsend and Lapata, 2011) do not distinguish between optional and obligatory arguments. For instance Zhu et al. (2010) simplifies (3C) to (3S) thereby incorrectly deleting the obligatory theme (*gifts*) of the complex sentence and modifying its meaning to *giving knights and warriors* (instead of *giving gifts to knights and warriors*).

(3) **C.** Women would also often give knights and warriors gifts that included thyme leaves as it was believed to bring courage to the bearer.
S. Women also often give knights and warriors. Gifts included thyme leaves as it was thought to bring courage to the saint. (Zhu et al., 2010)

We also depart from Coster and Kauchak (2011) who rely on null phrasal alignments for deletion during phrase based machine translation. In their approach, deletion is constrained by the training data and the possible alignments, independent of any linguistic knowledge.

Substitution and Reordering SMT based approaches to paraphrasing (Barzilay and Elhadad, 2003; Bannard and Callison-Burch, 2005) and to sentence simplification (Wubben et al., 2012) have shown that by utilising knowledge about alignment and translation probabilities, SMT systems can account for the substitutions and the reorderings occurring in sentence simplification. Following on these approaches, we therefore rely on phrase based SMT to learn substitutions and reordering. In addition, the language model we integrate in the SMT module helps ensuring better fluency and grammaticality.

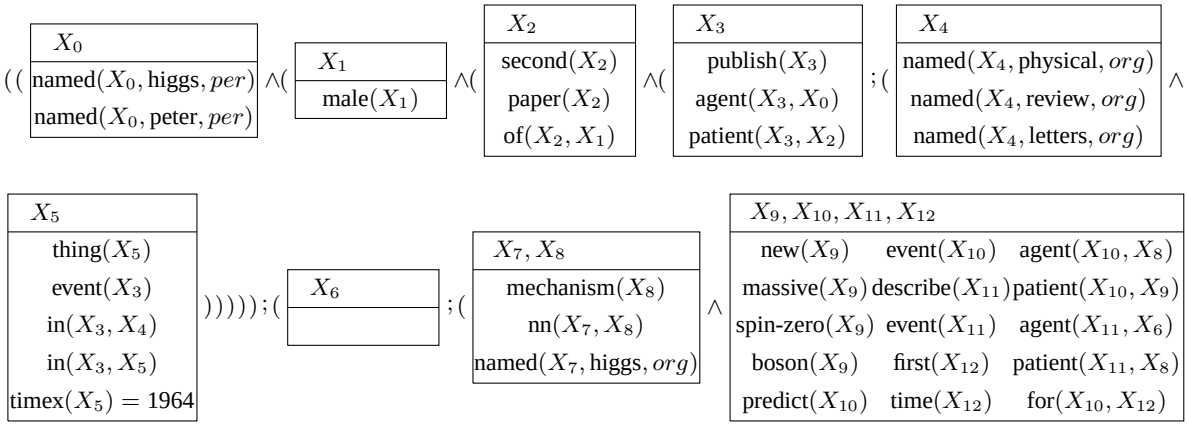
3.1 An Example

Figure 1 shows how our approach simplifies (4C) into (4S).

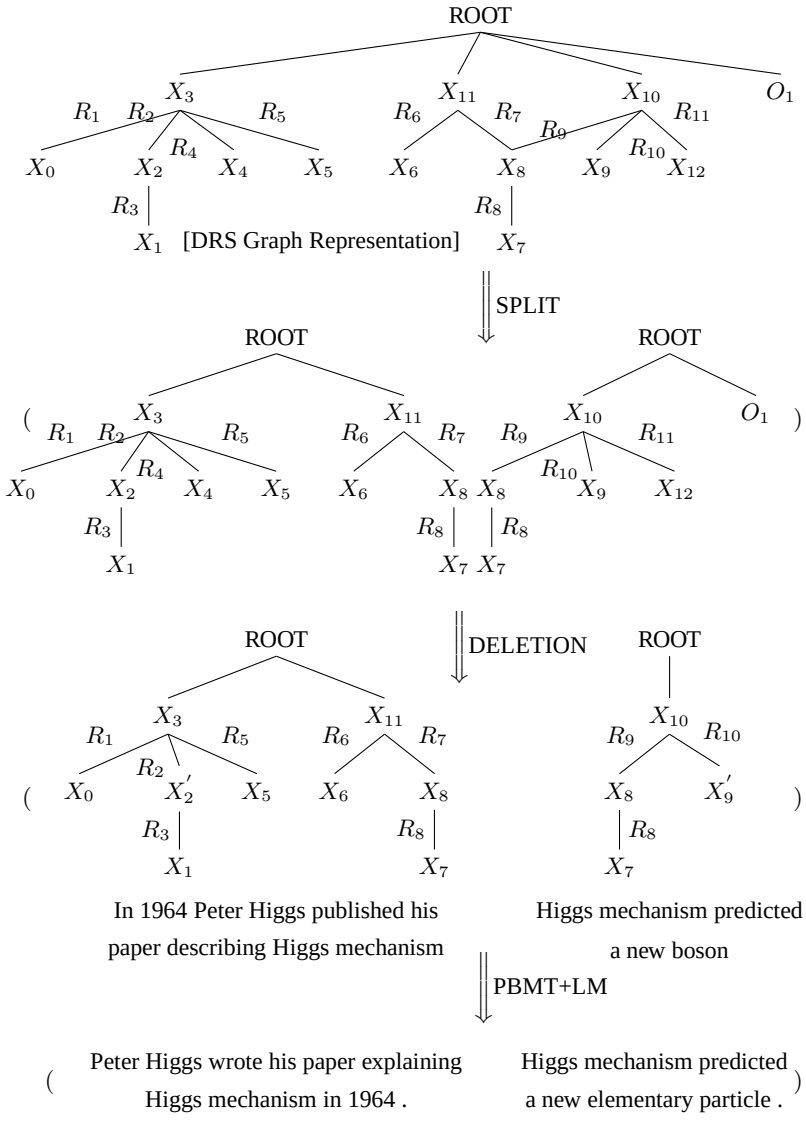
(4) **C.** In 1964 Peter Higgs published his second paper in Physical Review Letters describing Higgs mechanism which predicted a new massive spin-zero boson for the first time.
S. Peter Higgs wrote his paper explaining Higgs mechanism in 1964. Higgs mechanism predicted a new elementary particle.

The DRS for (4C) produced using Boxer (Curran et al., 2007) is shown at the top of the Figure and a graph representation[3] of the dependencies between its variables is shown immediately below. Each DRS variable labels a node in the graph and each edge is labelled with the relation holding between the variables labelling its end vertices. The

[3]The DRS to graph conversion goes through several preprocessing steps: the relation *nn* is inverted making modifier noun (*higgs*) dependent of modified noun (*mechanism*), *named* and *timex* are converted to unary predicates, e.g., $named(x, peter)$ is mapped to $peter(x)$ and $timex(x) = 1964$ is mapped to $1964(x)$; and nodes are introduced for orphan words (e.g., *which*).



[Discourse Representation Structure produced by BOXER]



node	pos. in S	predicate/type
X_0	3, 4	higgs/per, peter/per
X_1	6	male/a
X_2	6, 7, 8	second/a, paper/a
X_3	5	publish/v, **event**
X_4	10, 11, 12	physical/org review/org, letters/org
X_5	2	thing/n, 1964
X_6	6, 7, 8	$--$
X_7	14	higgs/org
X_8	14, 15	mechanism/n
X_9	18, 19, 20 21, 22	new/a, spin-zero/a massive/a, boson/n
X_{10}	17	predict/v, **event**
X_{11}	13	describe/v, **event**
X_{12}	24, 25, 26	first/a, time/n
O_1	16	which/WDT

rel	pos. in S	predicate
R_1	5	$agent, X_3 \rightarrow X_0$
R_2	5	$patient, X_3 \rightarrow X_2$
R_3	6	$of, X_2 \rightarrow X_1$
R_4	9	$in, X_3 \rightarrow X_4$
R_5	1	$in, X_3 \rightarrow X_5$
R_6	13	$agent, X_{11} \rightarrow X_6$
R_7	13	$patient, X_{11} \rightarrow X_8$
R_8	$--$	$nn, X_8 \rightarrow X_7$
R_9	17	$agent, X_{10} \rightarrow X_8$
R_{10}	17	$patient, X_{10} \rightarrow X_9$
R_{11}	23	$for, X_{10} \rightarrow X_{12}$

Figure 1: Simplification of "*In 1964 Peter Higgs published his second paper in Physical Review Letters describing Higgs mechanism which predicted a new massive spin-zero boson for the first time .*"

two tables to the right of the picture show the predicates (top table) associated with each variable and the relation label (bottom table) associated with each edge. Boxer also outputs the associated positions in the complex sentence for each predicate (not shown in the DRS but in the graph tables). Orphan words (OW) i.e., words which have no corresponding material in the DRS (e.g., *which* at position 16), are added to the graph (node O_1) thus ensuring that the position set associated with the graph exactly matches the positions in the input sentence and thus deriving the input sentence.

Split Candidate	isSplit	prob.
$(agent, for, patient)$ - $(agent, in, in, patient)$	true	0.63
	false	0.37

Table 1: Simplification: SPLIT

Given the input DRS shown in Figure 1, simplification proceeds as follows.

Splitting. The splitting candidates of a DRS are event pairs contained in that DRS. More precisely, the splitting candidates are pairs[4] of event variables associated with at least one of the core thematic roles (e.g., *agent* and *patient*). The features conditioning a split are the set of thematic roles associated with each event variable. The DRS shown in Figure 1 contains three such event variables X_3, X_{11} and X_{10} with associated thematic role sets {*agent, in, in, patient*}, {*agent, patient*} and {*agent, for, patient*} respectively. Hence, there are 3 splitting candidates (X_3-X_{11}, X_3-X_{10} and X_{10}-X_{11}) and 4 split options: no split or split at one of the splitting candidates. Here the split with highest probability (cf. Table 1) is chosen and the DRS is split into two sub-DRS, one containing X_3, and the other containing X_{10}. After splitting, dangling subgraphs are attached to the root of the new subgraph maximizing either proximity or position overlap. Here the graph rooted in X_{11} is attached to the root dominating X_3 and the orphan word O_1 to the root dominating X_{10}.

Deletion. The deletion model (cf. Table 2) regulates the deletion of relations and their associated subgraph; of adjectives and adverbs; and of orphan words. Here, the relations *in* between X_3 and X_4 and *for* between X_{10} and X_{12} are deleted resulting in the deletion of the phrases *"in Physical Review Letters"* and *"for the first time"* as well as the adjectives *second, massive, spin-zero* and the orphan word *which*.

Substitution and Reordering. Finally the translation and language model ensures that *published, describing* and *boson* are simplified to *wrote, explaining* and *elementary particle* respectively; and that the phrase *"In 1964"* is moved from the beginning of the sentence to its end.

3.2 The Simplification Model

Our simplification framework consists of a probabilistic model for splitting and dropping which we call DRS simplification model (DRS-SM); a phrase based translation model for substitution and reordering (PBMT); and a language model learned on Simple English Wikipedia (LM) for fluency and grammaticality. Given a complex sentence c, we split the simplification process into two steps. First, DRS-SM is applied to D_c (the DRS representation of the complex sentence c) to produce one or more (in case of splitting) intermediate simplified sentence(s) s'. Second, the simplified sentence(s) s' is further simplified to s using a phrase based machine translation system (PBMT+LM). Hence, our model can be formally defined as:

$$\begin{aligned}\hat{s} &= \operatorname*{argmax}_s p(s|c) \\ &= \operatorname*{argmax}_s p(s'|c)p(s|s') \\ &= \operatorname*{argmax}_s p(s'|\mathrm{D}_c)p(s'|s)p(s)\end{aligned}$$

where the probabilities $p(s'|\mathrm{D}_c)$, $p(s'|s)$ and $p(s)$ are given by the DRS simplification model, the phrase based machine translation model and the language model respectively.

To get the DRS simplification model, we combine the probability of splitting with the probability of deletion:

$$p(s'|\mathrm{D}_c) = \sum_{\theta: str(\theta(\mathrm{D}_c))=s'} p(\mathrm{D}_{split}|\mathrm{D}_c)p(\mathrm{D}_{del}|\mathrm{D}_{split})$$

where θ is a sequence of simplification operations and $str(\theta(\mathrm{D}_c))$ is the sequence of words associated with a DRS resulting from simplifying D_c using θ.

The probability of a splitting operation for a given DRS D_c is:

$$p(\mathrm{D}_{split}|\mathrm{D}_c) = \begin{cases} \textsc{split}(sp^{\text{true}}_{\text{cand}}), & \text{split at } sp_{\text{cand}} \\ \prod_{sp_{\text{cand}}} \textsc{split}(sp^{\text{false}}_{\text{cand}}), & \text{otherwise} \end{cases}$$

[4]The splitting candidates could be sets of event variables depending on the number of splits required. Here, we consider pairs for 2 splits.

relation candidate		isDrop	prob.
relation word	length range		
in	0-2	true false	0.22 0.72
in	2-5	true false	0.833 0.167

mod. cand.	isDrop	prob.
mod word		
new	true false	0.22 0.72
massive	true false	0.833 0.167

OW candidate		isDrop	prob.
orphan word	isBoundary		
and	true	true false	0.82 0.18
which	false	true false	0.833 0.167

Table 2: Simplification: DELETION (Relations, modifiers and OW respectively)

That is, if the DRS is split on the splitting candidate sp_{cand}, the probability of the split is then given by the *SPLIT* table (Table 1) for the *isSplit* value "true" and the split candidate sp_{cand}; else it is the product of the probability given by the *SPLIT* table for the *isSplit* value "false" for all split candidate considered for D_c. As mentioned above, the features used for determining the split operation are the role sets associated with pairs of event variables (cf. Table 1).

The deletion probability is given by three models: a model for relations determining the deletion of prepositional phrases; a model for modifiers (adjectives and adverbs) and a model for orphan words (Table 2). All three deletion models use the associated word itself as a feature. In addition, the model for relations uses the PP length-range as a feature while the model for orphan words relies on boundary information i.e., whether or not, the OW occurs at the associated sentence boundary.

$$p(\text{D}_{del}|\text{D}_{split}) = \prod_{\text{rel}_{\text{cand}}} \text{DEL}_{\text{rel}}(\text{rel}_{\text{cand}}) \prod_{\text{mod}_{\text{cand}}} \text{DEL}_{\text{mod}}(\text{mod}_{\text{cand}}) \prod_{\text{ow}_{\text{cand}}} \text{DEL}_{\text{ow}}(\text{ow}_{\text{cand}})$$

3.3 Estimating the parameters

We use the EM algorithm (Dempster et al., 1977) to estimate our split and deletion model parameters. For an efficient implementation of EM algorithm, we follow the work of Yamada and Knight (2001) and Zhu et al. (2010); and build training graphs (Figure 2) from the pair of complex and simple sentence pairs in the training data.

Each training graph represents a complex-simple sentence pair and consists of two types of nodes: major nodes (M-nodes) and operation nodes (O-nodes). An M-node contains the DRS representation D_c of a complex sentence c and the associated simple sentence(s) s_i while O-nodes determine split and deletion operations on their parent M-node. Only the root M-node is considered for the split operations. For example, given

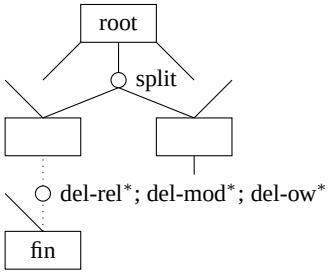


Figure 2: An example training graph

the root M-node $(\text{D}_c, (s_1, s_2))$, multiple successful split O-nodes will be created, each one further creating two M-nodes (D_{c1}, s_1) and (D_{c2}, s_2). For the training pair (c, s), the root M-node (D_c, s) is followed by a single split O-node producing an M-node (D_c, s) and counting all split candidates in D_c for failed split. The M-nodes created after split operations are then tried for multiple deletion operations of relations, modifiers and OW respectively. Each deletion candidate creates a deletion O-node marking successful or failed deletion of the candidate and a result M-node. The deletion process continues on the result M-node until there is no deletion candidate left to process. The governing criteria for the construction of the training graph is that, at each step, it tries to minimize the Levenshtein edit distance between the complex and the simple sentences. Moreover, for the splitting operation, we introduce a split only if the reference sentence consists of several sentences (i.e., there is a split in the training data); and only consider splits which maximises the overlap between split and simple reference sentences.

We initialize our probability tables Table 1 and Table 2 with the uniform distribution, i.e., 0.5 because all our features are binary. The EM algorithm iterates over training graphs counting model features from O-nodes and updating our probability tables. Because of the space constraints, we do not describe our algorithm in details. We refer the reader to (Yamada and Knight, 2001) for more details.

Our phrase based translation model is trained using the Moses toolkit[5] with its default command line options on the PWKP corpus (except the sentences from the test set) considering the complex sentence as the source and the simpler one as the target. Our trigram language model is trained using the SRILM toolkit[6] on the SWKP corpus[7].

Decoding. We explore the decoding graph similar to the training graph but in a greedy approach always picking the choice with maximal probability. Given a complex input sentence c, a split O-node will be selected corresponding to the decision of whether to split and where to split. Next, deletion O-nodes are selected indicating whether or not to drop each of the deletion candidate. The DRS associated with the final M-node D_{fin} is then mapped to a simplified sentence s'_{fin} which is further simplified using the phrase-based machine translation system to produce the final simplified sentence s_{simple}.

4 Experiments

We trained our simplification and translation models on the PWKP corpus. To evaluate performance, we compare our approach with three other state of the art systems using the test set provided by Zhu et al. (2010) and relying both on automatic metrics and on human judgments.

4.1 Training and Test Data

The DRS-Based simplification model is trained on PWKP, a bi-text of complex and simple sentences provided by Zhu et al. (2010). To construct this bi-text, Zhu et al. (2010) extracted complex and simple sentences from EWKP and SWKP respectively and automatically aligned them using TF*IDF as a similarity measure. PWKP contains $108016/114924$ complex/simple sentence pairs. We tokenize PWKP using Stanford CoreNLP toolkit[8]. We then parse all complex sentences in PWKP using Boxer[9] to produce their DRSs. Finally, our DRS-Based simplification model is trained on 97.75% of PWKP; we drop out 2.25% of the complex sentences in PWKP which are repeated in the test set or for which Boxer fails to produce DRSs.

We evaluate our model on the test set used by Zhu et al. (2010) namely, an aligned corpus of $100/131$ EWKP/SWKP sentences. Boxer produces a DRS for 96 of the 100 input sentences. These input are simplified using our simplification system namely, the DRS-SM model and the phrase-based machine translation system (Section 3.2). For the remaining four complex sentences, Boxer fails to produce DRSs. These four sentences are directly sent to the phrase-based machine translation system to produce simplified sentences.

4.2 Automatic Evaluation Metrics

To assess and compare simplification systems, two main automatic metrics have been used in previous work namely, BLEU and the Flesch-Kincaid Grade Level Index (FKG).

The FKG index is a readability metric taking into account the average sentence length in words and the average word length in syllables. In its original context (language learning), it was applied to well formed text and thus measured the simplicity of a well formed sentence. In the context of the simplification task however, the automatically generated sentences are not necessarily well formed so that the FKG index reduces to a measure of the sentence length (in terms of words and syllables) approximating the simplicity level of an output sentence irrespective of the length of the corresponding input. To assess simplification, we instead use metrics that are directly related to the simplification task namely, the number of splits in the overall (test and training) data and in average per sentences; the number of generated sentences with no edits i.e., which are identical to the original, complex one; and the average Levenshtein distance between the system's output and both the complex and the simple reference sentences.

BLEU gives a measure of how close a system's output is to the gold standard simple sentence. Because there are many possible ways of simplifying a sentence, BLEU alone fails to correctly assess the appropriateness of a simplification. Moreover BLEU does not capture the degree to which the system's output differs from the complex sentence input. We therefore use BLEU as a means to evaluate how close the systems output are to the reference corpus but complement it with further manual metrics capturing other important factors when

[5] http://www.statmt.org/moses/

[6] http://www.speech.sri.com/projects/srilm/

[7] We downloaded the snapshots of Simple Wikipedia dated 2013-10-30 available at http://dumps.wikimedia.org/.

[8] http://nlp.stanford.edu/software/corenlp.shtml

[9] http://svn.ask.it.usyd.edu.au/trac/candc, Version 1.00

evaluating simplifications such as the fluency and the adequacy of the output sentences and the degree to which the output sentence simplifies the input.

4.3 Results and Discussion

Number of Splits Table 3 shows the proportion of input whose simplification involved a splitting operation. While our system splits in proportion similar to that observed in the training data, the other systems either split very often (80% of the time for Zhu and 63% of the time for Woodsend) or not at all (Wubben). In other words, when compared to the other systems, our system performs splits in proportion closest to the reference both in terms of total number of splits and of average number of splits per sentence.

Data	Total number of sentences	% split	average split / sentence
PWKP	108,016	6.1	1.06
GOLD	100	28	1.30
Zhu	100	80	1.80
Woodsend	100	63	2.05
Wubben	100	1	1.01
Hybrid	100	10	1.10

Table 3: Proportion of Split Sentences (% split) in the training/test data and in average per sentence (average split / sentence). GOLD is the test data with the gold standard SWKP sentences; Zhu, Woodsend, Wubben are the best output of the models of Zhu et al. (2010), Woodsend and Lapata (2011) and Wubben et al. (2012) respectively; Hybrid is our model.

Number of Edits Table 4 indicates the edit distance of the output sentences w.r.t. both the complex and the simple reference sentences as well as the number of input for which no simplification occur. The right part of the table shows that our system generate simplifications which are closest to the reference sentence (in terms of edits) compared to those output by the other systems. It also produces the highest number of simplifications which are identical to the reference. Conversely our system only ranks third in terms of dissimilarity with the input complex sentences (6.32 edits away from the input sentence) behind the Woodsend (8.63 edits) and the Zhu (7.87 edits) system. This is in part due to the difference in splitting strategies noted above : the many splits applied by these latter two systems correlate with a high number of edits.

System	BLEU	Edits (Complex to System)		Edits (System to Simple)	
		LD	No edit	LD	No edit
GOLD	100	12.24	3	0	100
Zhu	37.4	7.87	2	14.64	0
Woodsend	42	8.63	24	16.03	2
Wubben	41.4	3.33	6	13.57	2
Hybrid	53.6	6.32	4	11.53	3

Table 4: Automated Metrics for Simplification: average Levenshtein distance (LD) to complex and simple reference sentences per system ; number of input sentences for which no simplification occur (No edit).

BLEU score We used Moses support tools: multi-bleu[10] to calculate BLEU scores. The BLEU scores shown in Table 4 show that our system produces simplifications that are closest to the reference.

In sum, the automatic metrics indicate that our system produces simplification that are consistently closest to the reference in terms of edit distance, number of splits and BLEU score.

4.4 Human Evaluation

The human evaluation was done online using the LG-Eval toolkit (Kow and Belz, 2012)[11]. The evaluators were allocated a trial set using a Latin Square Experimental Design (LSED) such that each evaluator sees the same number of output from each system and for each test set item. During the experiment, the evaluators were presented with a pair of a complex and a simple sentence(s) and asked to rate this pair w.r.t. to adequacy (Does the simplified sentence(s) preserve the meaning of the input?) and simplification (Does the generated sentence(s) simplify the complex input?). They were also asked to rate the second (simplified) sentence(s) of the pair w.r.t. to fluency (Is the simplified output fluent and grammatical?). Similar to the Wubben's human evaluation setup, we randomly selected 20 complex sentences from Zhu's test corpus and included in the evaluation corpus: the corresponding simple (Gold) sentence from Zhu's test corpus, the output of our system (Hybrid) and the output of the other three systems (Zhu, Woodsend and Wubben) which were provided to us by the system authors. The evaluation data thus consisted of 100 complex/simple pairs. We collected ratings from 27 participants.

[10] http://www.statmt.org/moses/?n=Moses.SupportTools
[11] http://www.nltg.brighton.ac.uk/research/lg-eval/

All were either native speakers or proficient in English, having taken part in a Master taught in English or lived in an English speaking country for an extended period of time.

Systems	Simplification	Fluency	Adequacy
GOLD	3.57	3.93	3.66
Zhu	2.84	2.34	2.34
Woodsend	1.73	2.94	3.04
Wubben	1.81	3.65	3.84
Hybrid	3.37	3.55	3.50

Table 5: Average Human Ratings for simplicity, fluency and adequacy

Table 5 shows the average ratings of the human evaluation on a slider scale from 0 to 5. Pairwise comparisons between all models and their statistical significance were carried out using a one-way ANOVA with post-hoc Tukey HSD tests and are shown in Table 6.

Systems	GOLD	Zhu	Woodsend	Wubben
Zhu	◊□△			
Woodsend	◊□△	◊□△		
Wubben	◊■▲	◊□△	♦□△	
Hybrid	♦■▲	♦□△	◊□▲	◊■▲

Table 6: ◊/♦ is/not significantly different (sig. diff.) wrt simplicity. □/■ is/not sig. diff. wrt fluency. △/▲ is/not sig. diff. wrt adequacy. (significance level: $p < 0.05$)

With regard to simplification, our system ranks first and is very close to the manually simplified input (the difference is not statistically significant). The low rating for Woodsend reflects the high number of unsimplified sentences (24/100 in the test data used for the automatic evaluation and 6/20 in the evaluation data used for human judgments). Our system data is not significantly different from the manually simplified data for simplicity whereas all other systems are.

For fluency, our system rates second behind Wubben and before Woodsend and Zhu. The difference between our system and both Zhu and Woodsend system is significant. In particular, Zhu's output is judged less fluent probably because of the many incorrect splits it licenses. Manual examination of the data shows that Woodsend's system also produces incorrect splits. For this system however, the high proportion of non simplified sentences probably counterbalances these incorrect splits, allowing for a good fluency score overall.

Regarding adequacy, our system is against closest to the reference (3.50 for our system vs. 3.66 for manual simplification). Our system, the Wubben system and the manual simplifications are in the same group (the differences between these systems are not significant). The Woodsend system comes second and the Zhu system third (the difference between the two is significant). Wubben's high fluency, high adequacy but low simplicity could be explained with their minimal number of edit (3.33 edits) from the source sentence.

In sum, if we group together systems for which there is no significant difference, our system ranks first (together with GOLD) for simplicity; first for fluency (together with GOLD and Wubben); and first for adequacy (together with GOLD and Wubben).

5 Conclusion

A key feature of our approach is that it is semantically based. Typically, discourse level simplification operations such as sentence splitting, sentence reordering, cue word selection, referring expression generation and determiner choice are semantically constrained. As argued by Siddharthan (2006), correctly capturing the interactions between these phenomena is essential to ensuring text cohesion. In the future, we would like to investigate how our framework deals with such discourse level simplifications i.e., simplifications which involves manipulation of the coreference and of the discourse structure. In the PWKP data, the proportion of split sentences is rather low (6.1 %) and many of the split sentences are simple sentence coordination splits. A more adequate but small corpus is that used in (Siddharthan, 2006) which consists of 95 cases of discourse simplification. Using data from the language learning or the children reading community, it would be interesting to first construct a similar, larger scale corpus; and to then train and test our approach on more complex cases of sentence splitting.

Acknowledgments

We are grateful to Zhemin Zhu, Kristian Woodsend and Sander Wubben for sharing their data. We would like to thank our annotators for participating in our human evaluation experiments and to anonymous reviewers for their insightful comments.

References

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL)*, pages 597–604. Association for Computational Linguistics.

Regina Barzilay and Noemie Elhadad. 2003. Sentence alignment for monolingual comparable corpora. In *Proceedings of the 2003 conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 25–32. Association for Computational Linguistics.

Stefan Bott, Horacio Saggion, and Simon Mille. 2012. Text simplification tools for spanish. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC)*, pages 1665–1671.

Yvonne Margaret Canning. 2002. *Syntactic simplification of Text*. Ph.D. thesis, University of Sunderland.

John Carroll, Guido Minnen, Darren Pearce, Yvonne Canning, Siobhan Devlin, and John Tait. 1999. Simplifying text for language-impaired readers. In *Proceedings of 9th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, volume 99, pages 269–270. Citeseer.

Raman Chandrasekar and Bangalore Srinivas. 1997. Automatic induction of rules for text simplification. *Knowledge-Based Systems*, 10(3):183–190.

Raman Chandrasekar, Christine Doran, and Bangalore Srinivas. 1996. Motivations and methods for text simplification. In *Proceedings of the 16th International conference on Computational linguistics (COLING)*, pages 1041–1044. Association for Computational Linguistics.

William Coster and David Kauchak. 2011. Learning to simplify sentences using wikipedia. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 1–9. Association for Computational Linguistics.

James R Curran, Stephen Clark, and Johan Bos. 2007. Linguistically motivated large-scale NLP with C&C and Boxer. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL) on Interactive Poster and Demonstration Sessions*, pages 33–36. Association for Computational Linguistics.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

Mark Dras. 1999. *Tree adjoining grammar and the reluctant paraphrasing of text*. Ph.D. thesis, Macquarie University NSW 2109 Australia.

Katja Filippova and Michael Strube. 2008. Dependency tree based sentence compression. In *Proceedings of the Fifth International Natural Language Generation Conference (INLG)*, pages 25–32. Association for Computational Linguistics.

Hans Kamp. 1981. A theory of truth and semantic representation. In J.A.G. Groenendijk, T.M.V. Janssen, B.J. Stokhof, and M.J.B. Stokhof, editors, *Formal methods in the study of language*, number pt. 1 in Mathematical Centre tracts. Mathematisch Centrum.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization-step one: Sentence compression. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI) and Twelfth Conference on Innovative Applications of Artificial Intelligence (IAAI)*, pages 703–710. AAAI Press.

Eric Kow and Anja Belz. 2012. LG-Eval: A Toolkit for Creating Online Language Evaluation Experiments. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC)*, pages 4033–4037.

Advaith Siddharthan, Ani Nenkova, and Kathleen McKeown. 2004. Syntactic simplification for improving content selection in multi-document summarization. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, page 896. Association for Computational Linguistics.

Advaith Siddharthan. 2002. An architecture for a text simplification system. In *Proceedings of the Language Engineering Conference (LEC)*, pages 64–71. IEEE Computer Society.

Advaith Siddharthan. 2006. Syntactic simplification and text cohesion. *Research on Language and Computation*, 4(1):77–109.

Advaith Siddharthan. 2010. Complex lexico-syntactic reformulation of sentences using typed dependency representations. In *Proceedings of the 6th International Natural Language Generation Conference (INLG)*, pages 125–133. Association for Computational Linguistics.

Advaith Siddharthan. 2011. Text simplification using typed dependencies: a comparison of the robustness of different generation strategies. In *Proceedings of the 13th European Workshop on Natural Language Generation (ENLG)*, pages 2–11. Association for Computational Linguistics.

David A Smith and Jason Eisner. 2006. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proceedings of the HLT-NAACL Workshop on Statistical Machine Translation*, pages 23–30. Association for Computational Linguistics.

David Vickrey and Daphne Koller. 2008. Sentence simplification for semantic role labeling. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL) and the Human Language Technology Conference (HLT)*, pages 344–352.

Willian Massami Watanabe, Arnaldo Candido Junior, Vinícius Rodriguez Uzêda, Renata Pontin de Mattos Fortes, Thiago Alexandre Salgueiro Pardo, and Sandra Maria Aluísio. 2009. Facilita: reading assistance for low-literacy readers. In *Proceedings of the 27th ACM international conference on Design of communication*, pages 29–36. ACM.

Kristian Woodsend and Mirella Lapata. 2011. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 409–420. Association for Computational Linguistics.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL): Long Papers-Volume 1*, pages 1015–1024. Association for Computational Linguistics.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics (ACL)*, pages 523–530. Association for Computational Linguistics.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*, pages 1353–1361, Stroudsburg, PA, USA. Association for Computational Linguistics.

Grammatical Relations in Chinese: GB-Ground Extraction and Data-Driven Parsing

Weiwei Sun, Yantao Du, Xin Kou, Shuoyang Ding, Xiaojun Wan*
Institute of Computer Science and Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{ws,duyantao,kouxin,wanxiaojun}@pku.edu.cn, dsy100@gmail.com

Abstract

This paper is concerned with building linguistic resources and statistical parsers for deep grammatical relation (GR) analysis of Chinese texts. A set of linguistic rules is defined to explore implicit phrase structural information and thus build high-quality GR annotations that are represented as general directed dependency graphs. The reliability of this linguistically-motivated GR extraction procedure is highlighted by manual evaluation. Based on the converted corpus, we study transition-based, data-driven models for GR parsing. We present a novel transition system which suits GR graphs better than existing systems. The key idea is to introduce a new type of transition that reorders top k elements in the memory module. Evaluation gauges how successful GR parsing for Chinese can be by applying data-driven models.

1 Introduction

Grammatical relations (GRs) represent functional relationships between language units in a sentence. They are exemplified in traditional grammars by the notions of subject, direct/indirect object, etc. GRs have assumed an important role in linguistic theorizing, within a variety of approaches ranging from generative grammar to functional theories. For example, several computational grammar formalisms, such as Lexical Function Grammar (LFG; Bresnan and Kaplan, 1982; Dalrymple, 2001) and Head-driven Phrase Structure Grammar (HPSG; Pollard and Sag, 1994) encode grammatical functions directly. In particular, GRs can be viewed as the dependency backbone of an LFG analysis that provide general linguistic insights, and have great potential advantages for NLP applications, (Kaplan et al., 2004; Briscoe and Carroll, 2006; Clark and Curran, 2007a; Miyao et al., 2007).

In this paper, we address the question of analyzing Chinese sentences with deep GRs. To acquire high-quality GR corpus, we propose a linguistically-motivated algorithm to translate a Government and Binding (GB; Chomsky, 1981; Carnie, 2007) grounded phrase structure treebank, i.e. Chinese Treebank (CTB; Xue et al., 2005) to a deep dependency bank where GRs are explicitly represented. Different from popular *shallow* dependency parsing that focus on tree-shaped structures, our GR annotations are represented as general directed graphs that express not only local but also various long-distance dependencies, such as coordinations, control/raising constructions, topicalization, relative clauses and many other complicated linguistic phenomena that goes beyond shallow syntax (see Fig. 1 for example.). Manual evaluation highlights the reliability of our linguistically-motivated GR extraction algorithm: The overall dependency-based precision and recall are 99.17 and 98.87. The automatically-converted corpus would be of use for a wide variety of NLP tasks.

Recent years have seen the introduction of a number of treebank-guided statistical parsers capable of generating considerably accurate parses for Chinese. With the high-quality GR resource at hand, we study data-driven GR parsing. Previous work on dependency parsing mainly focused on structures that can be represented in terms of directed trees. We notice two exceptions. Sagae and Tsujii (2008) and Titov et al. (2009) individually studied two transition systems that can generate more general graphs rather than trees. Inspired by their work, we study transition-based models for building deep dependency structures. The existence of a large number of crossing arcs in GR graphs makes left-to-right, incremental graph spanning computationally hard. Applied to our data, the two existing systems cover only 51.0% and 76.5% GR graphs respectively. To better suit

*Email correspondence.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 446–456,
Baltimore, Maryland, USA, June 23-25 2014.

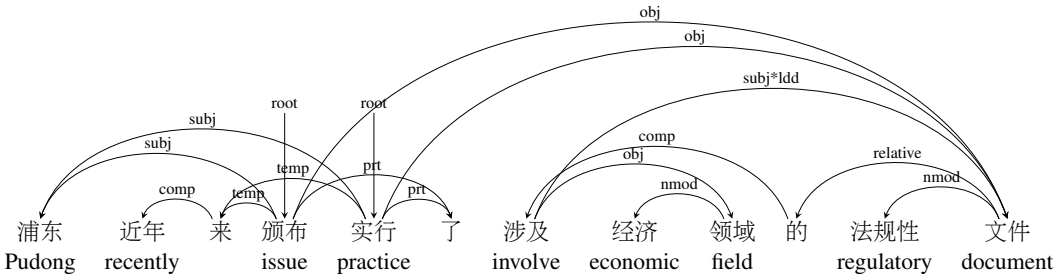


Figure 1: An example: *Pudong recently enacted regulatory documents involving the economic field*. The symbol "*ldd" indicates long-distance dependencies; "subj*ldd" between the word "涉及/involve" and the word "文件/documents" represents a long-range subject-predicate relation. The arguments and adjuncts of the coordinated verbs, namely "颁布/issue" and "实行/practice," are separately yet distributively linked the two heads.

our problem, we extend Titov et al.'s work and study what we call K-permutation transition system. The key idea is to introduce a new type of transition that reorders top k ($2 \leq k \leq K$) elements in the memory module of a stack-based transition system. With the increase of K, the expressiveness of the corresponding system strictly increases. We propose an oracle deriving method which is guaranteed to find a sound transition sequence if one exits. Moreover, we introduce an effective approximation of that oracle, which decreases decoding ambiguity but practically covers almost exactly the same graphs for our data.

Based on the stronger transition system, we build a GR parser with a discriminative model for disambiguation and a beam decoder for inference. We conduct experiments on CTB 6.0 to profile this parser. With the increase of the K, the parser is able to utilize more GR graphs for training and the numeric performance is improved. Evaluation gauges how successful GR parsing for Chinese can be by applying data-driven models. Detailed analysis reveal some important factors that may possibly boost the performance. To our knowledge, this work provides the first result of extensive experiments of parsing Chinese with GRs.

We release our GR processing kit and gold-standard annotations for research purposes. These resources can be downloaded at http://www.icst.pku.edu.cn/lcwm/omg.

2 GB-grounded GR Extraction

In this section, we discuss the construction of the GR annotations. Basically, the annotations are automatically converted from a GB-grounded phrase-structure treebank, namely CTB. Conceptually, this conversion is similar to the conversions from CTB structures to representations in deep grammar formalisms (Tse and Curran, 2010; Yu et al., 2010; Guo et al., 2007; Xia, 2001). However, our work is grounded in GB, which is the linguistic basis of the construction of CTB. We argue that this theoretical choice makes the conversion process more compatible with the original annotations and therefore more accurate. We use directed graphs to explicitly encode bi-lexical dependencies involved in coordination, raising/control constructions, extraction, topicalization, and many other complicated phenomena. Fig. 1 shows an example of such a GR graph and its original CTB annotation.

2.1 Linguistic Basis

GRs are encoded in different ways in different languages. In some languages, e.g. Turkish, grammatical function is encoded by means of morphological marking, while in highly *configurational* languages, e.g. Chinese, the grammatical function of a phrase is heavily determined by its constituent structure position. Dominant Chomskyan theories, including GB, have defined GRs as configurations at phrase structures. Following this principle, CTB groups words into constituents through the use of a limited set of fundamental grammatical functions. Transformational grammar utilizes empty categories (ECs) to represent long-distance dependencies. In CTB, traces are provided by relating displaced linguistic material to where it should be interpreted semantically. By exploiting configurational information, traces and functional tag annotations, GR information can be hopefully

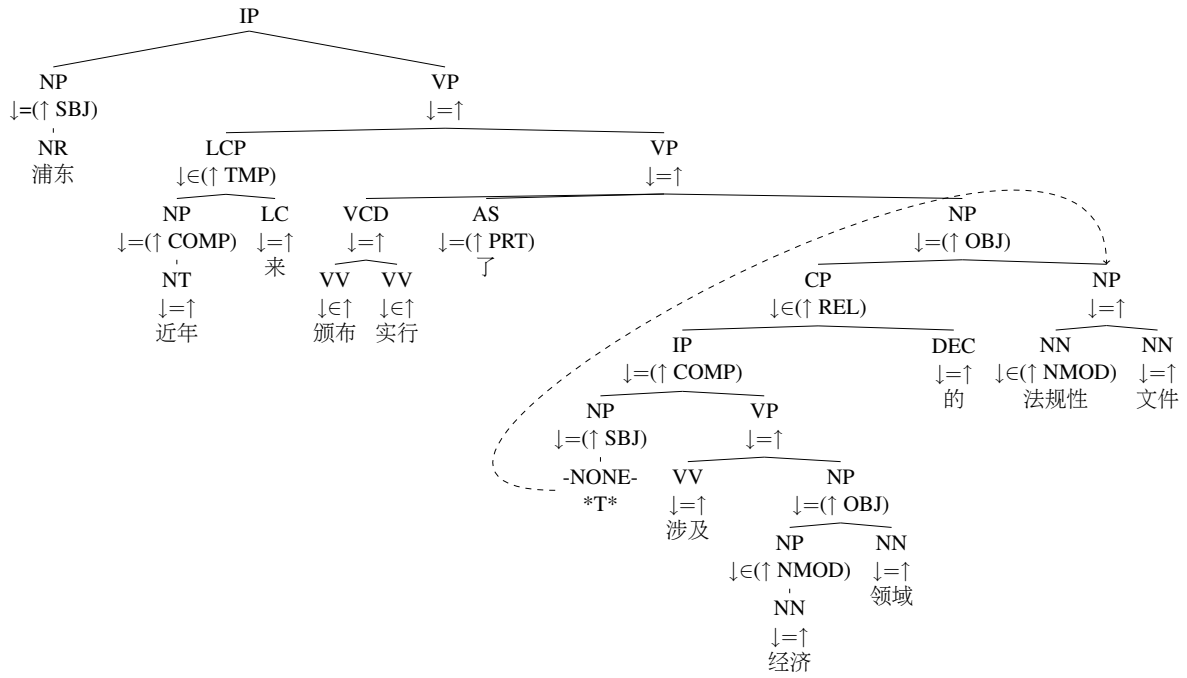


Figure 2: The original CTB annotation augmented with LFG-like f-structure annotations of the running example.

derived from CTB trees with high accuracy.

2.2 The Extraction Algorithm

Our treebank conversion algorithm borrows key insights from Lexical Functional Grammar (LFG; Bresnan and Kaplan, 1982; Dalrymple, 2001). LFG posits two levels of representation: c(onstituent)-structure and f(unctional)-structure minimally. C-structure is represented by phrase-structure trees, and captures surface syntactic configurations such as word order, while f-structure encodes grammatical functions. It is easy to extract a dependency backbone which approximates basic predicate-argument-adjunct structures from f-structures. The construction of the widely used PARC DepBank (King et al., 2003) is a good example.

LFG relates c-structure and f-structure through f-structure annotations, which compositionally map every constituent to a corresponding f-structure. Borrowing this key idea, we translate CTB trees to dependency graphs by first augmenting each constituency with f-structure annotations, then propagating the head words of the head or conjunct daughter(s) upwards to their parents, and finally creating a dependency graph. The following presents details step-by-step.

Tapping implicit information. Xue (2007) introduced a systematic study to tap the implicit functional information of CTB. This gives us a very good start to extract GRs. We slightly modify their method to enrich a CTB tree with f-structure annotations: Each node in a resulting tree is annotated with one and only one corresponding equation. See Fig. 2 for example. Comparing the original annotation and enriched one, we can see that the functionality of this step is to explicitly represent and regulate grammatical functions.

Beyond CTB annotations: tracing more. Natural languages do not always interpret linguistic material locally. In order to obtain accurate and complete GR, predicate-argument, or logical form representations, a hallmark of deep grammars is that they usually involve a non-local dependency resolution mechanism. CTB trees utilize ECs and coindexed materials to represent long-distance dependencies. An EC is a nominal element that does not have any phonological content and is therefore unpronounced. Two kinds of anaphoric ECs, i.e. big PRO and trace, are annotated in CTB. Theoretically speaking, only trace is generated as the result of *movement* and therefore annotated with antecedents in CTB. We carefully check the annotation and find that a considerable amount of antecedents are not labeled, and hence a lot of impor-

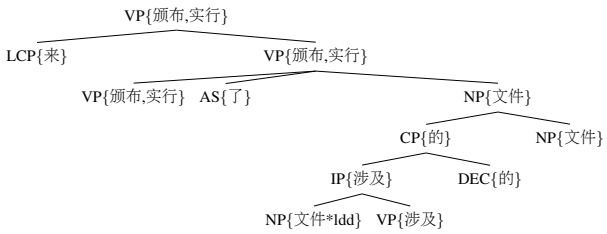


Figure 3: An example of lexicalized tree after head word upward passing. Only partial result is shown. The long-distance dependency between "涉及/involve" and "文件/document" is created through copying the dependent to a coindexed anaphoric EC position.

tant non-local information is missing. In addition, since the big PRO is also anaphoric, it is possible to find coindexed components sometimes. Such non-local information is also very valuable.

Beyond CTB annotations, we introduce a number of phrase-structure patterns to extract more non-local dependencies. The method heavily leverages linguistic rules to exploit structural information. We take into account both theoretical assumptions and analyzing practices to enrich coindexation information according to phrase-structure patterns. In particular, we try to link an anaphoric EC e with its c-commonders if no non-empty antecedent has already been coindexed with e. Because the CTB is influenced deeply by the X-bar syntax, which regulates constituent analysis much, the number of our linguistic rules is quite modest. For the development of conversion rules, we used the first 9 files of CTB, which contains about 100 sentences. Readers can refer to the well-documented Perl script for details. See Fig. 2 for example. The noun phrase "法规性文件/regulatory documents" is related to the trace "*T*." This coindexation is not labeled by the original annotation.

Passing head words and linking ECs. Based on an enriched tree, our algorithm propagates the head word of the head daughter upwards to their parents, linking coindexed units, and finally creating a GR graph. The partial result after head word passing of the running example is shown in Fig. 3. There are two differences of the head word passing between our GR extraction and a "normal" dependency tree extraction. First, the GR extraction procedure may pass multiple head words to its parent, especially in a coordination construction. Second, long-distance dependencies are created by linking ECs and their coindexed phrases.

	Precision	Recall	F-score
Unlabeled	99.48	99.17	99.32
Labeled	99.17	98.87	99.02

Table 1: Manual evaluation of 209 sentences.

2.3 Manual Evaluation

To have a precise understanding of whether our extraction algorithm works well, we have selected 20 files that contains 209 sentences in total for manual evaluation. Linguistic experts carefully examine the corresponding GR graphs derived by our extraction algorithm and correct all errors. In other words, a *gold standard* GR annotation set is created. The measure for comparing two dependency graphs is precision/recall of GR tokens which are defined as $\langle w_h, w_d, l \rangle$ tuples, where w_h is the head, w_d is the dependent and l is the relation. Labeled precision/recall (LP/LR) is the ratio of tuples correctly identified by the automatic generator, while unlabeled precision/recall (UP/UR) is the ratio regardless of l. F-score is a harmonic mean of precision and recall. These measures correspond to attachment scores (LAS/UAS) in dependency tree parsing. To evaluate our GR parsing models that will be introduced later, we also report these metrics.

The overall performance is summarized in Tab. 1. We can see that the automatical GR extraction achieves relatively high performance. There are two sources of errors in treebank conversion: (1) inadequate conversion rules and (2) wrong or inconsistent original annotations. During the creation of the gold standard corpus, we find that the former is mainly caused by complicated unbounded dependencies and the lack of internal structure for some kinds of phrases. Such problems are very hard to solve through rules only, if not possible, since original annotations do not provide sufficient information. The latter problem is more scattered and unpredictable.

2.4 Statistics

Allowing non-projective dependencies generally makes parsing either by graph-based or transition-based dependency parsing harder. Substantial research effort has been devoted in recent years to the design of elegant solutions for this problem. There are much more crossing arcs in the GR

graphs than syntactic dependency trees. In the training data (defined in Section 4.1), there are 558132 arcs and 86534 crossing pairs, About half of the sentences have crossing arcs (10930 out of 22277). The wide existence of crossing arcs poses an essential challenge for GR parsing, namely, to find methods for handling crossing arcs without a significant loss in accuracy and efficiency.

3 Transition-based GR Parsing

The availability of large-scale treebanks has contributed to the blossoming of statistical approaches to build accurate *shallow* constituency and dependency parsers. With high-quality GR resources at hand, it is possible to study statistical approaches to automatically parse GR graphs. In this section, we investigate the feasibility of applying a data-driven, grammar-free approach to build GRs directly. In particular, transition-based dependency parsing method is studied.

3.1 Data-Driven Dependency Parsing

Data-driven, grammar-free dependency parsing has received an increasing amount of attention in the past decade. Such approaches, e.g. transition-based (Yamada and Matsumoto, 2003; Nivre, 2008) and graph-based (McDonald, 2006; Torres Martins et al., 2009) models have attracted the most attention of dependency parsing in recent years. Transition-based parsers utilize transition systems to derive dependency trees together with treebank-induced statistical models for predicting transitions. This approach was pioneered by (Yamada and Matsumoto, 2003) and (Nivre et al., 2004). Most research concentrated on surface syntactic structures, and the majority of existing approaches are limited to producing only trees. We notice two exceptions. Sagae and Tsujii (2008) and Titov et al. (2009) individually introduced two transition systems that can generate specific graphs rather than trees. Inspired by their work, we study transition-based approach to build GR graphs.

3.2 Transition Systems

Following (Nivre, 2008), we define a transition system for dependency parsing as a quadruple $S = (\mathcal{C}, T, c_s, \mathcal{C}_t)$, where

1. $\mathcal{C}$ is a set of configurations, each of which contains a buffer β of (remaining) words and a set A of dependency arcs,

Transitions	
SHIFT	$(\sigma, j\|\beta, A) \Rightarrow (\sigma\|j, \beta, A)$
LEFT-ARC$_l$	$(\sigma\|i, j\|\beta, A) \Rightarrow (\sigma\|i, j\|\beta, A \cup \{(j, l, i)\})$
RIGHT-ARC$_l$	$(\sigma\|i, j\|\beta, A) \Rightarrow (\sigma\|i, j\|\beta, A \cup \{(i, l, j)\})$
POP	$(\sigma\|i, \beta, A) \Rightarrow (\sigma, \beta, A)$
ROTATE$_k$	$(\sigma\|i_k]\dots\|i_2\|i_1, \beta, A) \Rightarrow (\sigma\|i_1\|i_k]\dots\|i_2, \beta, A)$

Table 2: K-permutation System.

2. T is a set of transitions, each of which is a (partial) function $t : \mathcal{C} \mapsto \mathcal{C}$,

3. c_s is an initialization function, mapping a sentence x to a configuration, with $\beta = [1, \dots, n]$,

4. $\mathcal{C}_t \subseteq \mathcal{C}$ is a set of terminal configurations.

Given a sentence $x = w_1, \dots, w_n$ and a graph $G = (V, A)$ on it, if there is a sequence of transitions $t_1, \dots, t_m$ and a sequence of configurations $c_0, \dots, c_m$ such that $c_0 = c_s(x)$, $t_i(c_{i-1}) = c_i (i = 1, \dots, m)$, $c_m \in \mathcal{C}_t$, and $A_{c_m} = A$, we say the sequence of transitions is an *oracle* sequence. And we define $\bar{A}_{c_i} = A - A_{c_i}$ for the arcs to be built in c_i. In a typical transition-based parsing process, the input words are put into a queue and partially built structures are organized by a stack. A set of SHIFT/REDUCE actions are performed sequentially to consume words from the queue and update the partial parsing results.

3.3 Online Reordering

Among existing systems, Sagae and Tsujii's is designed for projective graphs (denoted by $\mathcal{G}_1$ in Definition 1), and Titov et al.'s handles only a specific subset of non-projective graphs as well as projective graphs ($\mathcal{G}_2$). Applied to our data, only 51.0% and 76.5% of the extracted graphs are parsable with their systems. Obviously, it is necessary to investigate new transition systems for the parsing task in our study. To deal with crossing arcs, Titov et al. (2009) and Nivre (2009) designed a SWAP transition that switches the position of the two topmost nodes on the stack. Inspired by their work, we extend this approach to parse more general graphs. The basic idea is to provide our new system with an ability to reorder more nodes during decoding in an online fashion, which we refer to as online reordering.

3.4 K-Permutation System

We define a K-permutation transition system $S_K = (\mathcal{C}, T, c_s, \mathcal{C}_t)$, where a configuration $c =$

$(\sigma, \beta, A) \in \mathcal{C}$ contains a stack σ of nodes besides β and A. We set the initial configuration for a sentence $x = w_1, \ldots, w_n$ to be $c_s(x) = ([], [1, \ldots, n], \{\})$, and take $\mathcal{C}_t$ to be the set of all configurations of the form $c_t = (\sigma, [], A)$ (for any arc set A). The set of transitions T contains five types of actions, as shown in Tab. 2:

1. SHIFT removes the front element from β and pushes it onto σ.
2. LEFT-ARC$_l$/RIGHT-ARC$_l$ updates a configuration by adding (i, l, j)/(j, l, i) to A where i is the top of σ, and j is the front of β.
3. POP deletes the top element of σ.
4. ROTATE$_k$ updates a configuration with stack $\sigma|i_k|\ldots|i_2|i_1$ by rotating the top k nodes in stack left by one index, obtaining $\sigma|i_1|i_k|\ldots|i_2$, with constraint $2 \leq k \leq K$.

We refer to this system as K-permutation because by rotating the top k $(2 \leq k \leq K)$ nodes in the stack, we can obtain all the permutations of the top K nodes. Note that S_2 is identical to Titov et al.'s; S_∞ is complete with respect to the class of all directed graphs without self-loop, since we can arbitrarily permute the nodes in the stack. The K-permutation system exhibits a nice property: The sets of corresponding graphs are strictly monotonic with respect to the $\subset$ operation.

Definition 1. If a graph G can be parsed with transition system S_K, we say G is a *K-perm graph*. We use $\mathcal{G}_K$ to denote the set of all k-perm graphs. Specially, $\mathcal{G}_0 = \emptyset$, $\mathcal{G}_1$ is the set of all projective graphs, and $\mathcal{G}_\infty = \bigcup_{k=0}^{\infty} \mathcal{G}_k$.

Theorem 1. $\mathcal{G}_i \subsetneq \mathcal{G}_{i+1}, \forall i \geq 0$.

Proof. It is obvious that $\mathcal{G}_i \subseteq \mathcal{G}_{i+1}$ and $\mathcal{G}_0 \subsetneq \mathcal{G}_1$. Fig. 4 gives an example which is in $\mathcal{G}_{i+1}$ but not in $\mathcal{G}_i$ for all $i > 0$, indicating $\mathcal{G}_i \neq \mathcal{G}_{i+1}$. □

Theorem 2. $\mathcal{G}_\infty$ is the set of all graphs without self-loop.

Proof. It follows immediately from the fact that $G \in \mathcal{G}_{|V|}, \forall G = \langle V, E \rangle$. □

The transition systems introduced in (Sagae and Tsujii, 2008) and (Titov et al., 2009) can be viewed as S_1[1] and S_2.

[1]Though Sagae and Tsujii (2008) introduced additional constraints to exclude cyclic path, the fundamental transition mechanism of their system is the same to S_1.

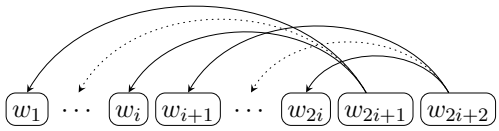


Figure 4: A graph which is in $\mathcal{G}_{i+1}$, but not in $\mathcal{G}_i$.

3.5 Normal Form Oracle

The K-permutation transition system may allow multiple oracle transition sequences on one graph, but trying to sum all the possible oracles is usually computational expensive. Here we give a construction procedure which is guaranteed to find an oracle sequence if one exits. We refer it as normal form oracle (NFO).

Let $L(j)$ be the ordered list of nodes connected to j in $\bar{A}_{c_{i-1}}$ for $j \in \sigma_{c_{i-1}}$, and let $\mathcal{L}_K(\sigma_{c_{i-1}}) = [L(j_1), \ldots, L(j_{\max\{l,K\}})]$. If $\sigma_{c_{i-1}}$ is empty, then we set t_i to SHIFT; if there is no arc linked to j_1 in $\bar{A}_{c_{i-1}}$, then we set t_i to POP; if there exits $a \in \bar{A}_{c_{i-1}}$ linking j_1 and b, then we set t_i to LEFT-ARC or RIGHT-ARC correspondingly. When there are only SHIFT and ROTATE left, we first apply a sequence of ROTATE's to make $\mathcal{L}_K(\sigma)$ complete ordered by lexicographical order, then apply a SHIFT. Let $c_i = t_i(c_{i-1})$, we continue to compute t_{i+1}, until β_{c_i} is empty.

Theorem 3. If a graph is parsable with the transition system S_K then the construction procedure is guaranteed to find an oracle transition sequence.

Proof. During the construction, all the arcs are built by LEFT-ARC or RIGHT-ARC, which links the top of the stack and the front of the buffer. Therefore, we prefer $\mathcal{L}(\sigma)$ to be as orderly as possible, to make the words to be linked sooner on the top of the stack. the construction procedure above does best within the power of the system S_K. □

3.6 An Approximation for NFO

In the construction of NFO transitions, we exhaustively use the ROTATE's to make $\mathcal{L}(\sigma)$ complete ordered. We also observed that the transition LEFT-ARC, RIGHT-ARC and SHIFT only change the relative order between the first element of $\mathcal{L}(\sigma)$ and the rest elements. Therefore we explored an approximate procedure to determine the ROTATE's, based on the observation. We call it approximate NFO (ANFO). Using notation defined in Section 3.5, the approximate procedure goes as follows. When it comes to the determination of

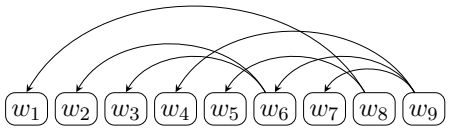


Figure 5: A graph that can be parsed with S_3 with a transition sequence $\text{SSSSR}_3\text{SR}_3\text{APAP}\mathbf{R}_2\mathbf{R}_3\text{SR}_3\text{SR}_3\text{APAPAPAPAP}$, where S stands for SHIFT, R for ROTATE, A for LEFT-ARC, and P for POP. But the approximate procedure fails to find the oracle, since $\mathbf{R}_2\mathbf{R}_3$ in bold in the sequence are not to be applied.

the ROTATE sequence, let k be the largest m such that $0 \leq m \leq \min\{K, l\}$ and $L(j_m)$ strictly precedes $L(j_1)$ by the lexicographical order (here we assume $L(j_0)$ strictly precedes any $L(j), j \in \sigma$). If $k > 0$, we set t_i to ROTATE_k; else we set t_i to SHIFT. The approximation assumes $\mathcal{L}(\sigma)$ is completely ordered except the first element, and insert the first element to its proper place each time.

Definition 2. We define $\hat{\mathcal{G}}_K$ as the graphs the oracle of which can be extracted by S_K with the approximation procedure.

It can be inferred similarly that Theorem 1 and Theorem 2 also hold for $\hat{\mathcal{G}}$'s. However, the $\hat{\mathcal{G}}_K$ is not equal to $\mathcal{G}_K$ in non-trivial cases.

Theorem 4. $\hat{\mathcal{G}}_i \subsetneq \mathcal{G}_i, \forall i \geq 3$.

Proof. It is trivial that $\hat{\mathcal{G}}_i \subseteq \mathcal{G}_i$. An example graph that is in $\mathcal{G}_3$ but not in $\hat{\mathcal{G}}_3$ is shown in Figure 5, examples for arbitrary $i > 3$ can be constructed similarly. □

The above theorem indicates the inadequacy of the ANFO deriving procedure. Nevertheless, empirical evaluation (Section 4.2) shows that the coverage of AFO and ANFO deriving procedures are almost identical when applying to linguistic data.

3.7 Statistical Parsing

When we parse a sentence $w_1 w_2 \cdots w_n$, we start with the initial configuration $c_0 = c_s(x)$, and choose next transition $t_i = C(c_{i-1})$ iteratively according to a discriminative classifier trained on oracle sequences. To build a parser, we use a structured classifier to approximate the oracle, and apply the Passive-Aggressive (PA) algorithm (Crammer et al., 2006) for parameter estimation. The PA algorithm is similar to the Perceptron algorithm, the difference from which is the update of weight vector. We also use parameter averaging and early update to achieve better training. Developing features has been shown crucial to advancing the state-of-the-art in dependency tree parsing (Koo and Collins, 2010; Zhang and Nivre, 2011). To build accurate deep dependency parsers, we utilize a large set of features for disambiguation. See the notes included in the supplymentary material for details. To improve the performance, we also apply the technique of beam search, which keep a beam of transition sequences with highest scores when parsing.

4 Experiments

4.1 Experimental setup

CTB is a segmented, part-of-speech (POS) tagged, and fully bracketed corpus in the constituency formalism, and very popular to evaluate fundamental NLP tasks, including word segmentation (Sun and Xu, 2011), POS tagging (Sun and Uszkoreit, 2012), and syntactic parsing (Zhang and Clark, 2009; Sun and Wan, 2013). We use CTB 6.0 and define the training, development and test sets according to the CoNLL 2009 shared task. We use gold-standard word segmentation and POS tagging results as inputs. All transition-based parsing models are trained with beam 16 and iteration 30. Overall precision/recall/f-score with respect to dependency tokens is reported. To evaluate the ability to recover non-local dependencies, the recall of such dependencies are reported too.

4.2 Coverage and Accuracy

There is a dual effect of the increase of the parameter k to our transition-based dependency parser. On one hand, the higher k is, the more expressivity the corresponding transition system has. A system with higher k covers more structures and allows to use more data for training. On the other hand, higher k brings more ambiguities to the corresponding parser, and the parsing performance may thus suffer. Note that the ambiguity exists not only in each step for transition decision, but also in selecting the training oracle.

The left-most columns of Tab. 3 shows the coverage of K-permutation transition system with respect to different K and different oracle deriving algorithms. Readers may be surprised that the coverage of NFO and ANFO deriving procedures is the same. Actually, all the covered graphs by the two oracle deriving procedures are exactly the

System	NFO	ANFO	UP	UR	UF	LP	LR	LF	UR_L	LR_L	UR_{NL}	LR_{NL}
S_2	76.5	76.5	85.88	81.00	83.37	83.98	79.21	81.53	81.93	80.34	58.88	52.17
S_3	89.0	89.0	86.02	81.72	83.82	84.07	79.86	81.91	82.61	80.94	60.46	54.28
S_4	95.6	95.6	86.28	82.06	84.12	84.35	80.22	82.23	82.92	81.29	61.48	54.77
S_5	98.4	98.4	86.44	82.21	84.27	84.51	80.37	82.39	83.15	81.51	59.80	53.30

Table 3: Coverage and accuracy of the GR parser on the development data.

same, except for S_3. Only 1 from 22277 sentences can find a NFO but not an ANFO. This number demonstrates the effectiveness of ANFO. In the following experiments, we use the ANFO's to train our parser.

Applied to our data, S_2, i.e. the exact system introduced by Titov et al. (2009), only covers 76.5% GR graphs. This is very different from the result obtained on the CoNLL shared task data for English semantic role labeling (SRL). According to (Titov et al., 2009), 99% semantic-role-labelled graphs can be generated by S_2. We think there are two main reasons accounting for the differences, and highlight the importance of the expressiveness of transition systems to solve deep dependency parsing problems. First, the SRL task only focuses on finding arguments and adjuncts of verbal (and nominal) predicates, while dependencies headed by other words are not contained in its graph representation. On contrast, a deep dependency structure, like GR graph, approximates deep syntactic or semantic information of a sentence as a whole, and therefore is more dense. As a result, permutation system with a very low k is incapable to handle more cases. Another reason is about the Chinese language. Some language-specific properties result in complex crossing arcs. For example, serial verb constructions are widely used in Chinese to describe several separate events without conjunctions. The verbal heads in such constructions share subjects and adjuncts, both of which are before the heads. The distributive dependencies between verbal heads and subjects/adjuncts usually produce crossing arcs (see Fig. 6). To test our assumption, we evaluate the coverage of S_2 over the functor-argument dependency graphs provided by the English and Chinese CCGBank (Hockenmaier and Steedman, 2007; Tse and Curran, 2010). The result is **96.9%** vs. **89.0%**, which confirms our linguistic intuition under another grammar formalism.

Tab. 3 summarizes the performance of the transition-based parser with different configurations to reveal how well data-driven parsing can

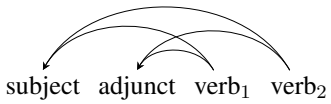


Figure 6: A simplified example to illustrate crossing arcs in serial verbal constructions.

be performed in realistic situations. We can see that with the increase of K, the overall parsing accuracy incrementally goes up. The high complexity of Chinese deep dependency structures demonstrates the importance of the expressiveness of a transition system, while the improved numeric accuracies practically certify the benefits. The two points merit further exploration to more expressive transition systems for deep dependency parsing, at least for Chinese. The labeled evaluation scores on the final test data are presented in Tab. 4.

Test	UP	UR	UF	LR_L	LR_{NL}
S_5	83.93	79.82	81.82	80.94	54.38

Table 4: Performance on the test data.

4.3 Precision vs. Recall

A noteworthy thing about the overall performance is that the precision is promising but the recall is too low behind. This difference is consistent with the result obtained by a shift-reduce CCG parser (Zhang and Clark, 2011). The functor-argument dependencies generated by that parser also has a relatively high precision but considerably low recall. There are two similarities between our parser and theirs: 1) both parsers produce dependency graphs rather trees; 2) both parser employ a beam decoder that does not guarantee global optimality. To build NLP application, e.g. information extraction, systems upon GR parsing, such property merits attention. A good trade-off between the precision and the recall may have a great impact on final results.

4.4 Local vs. Non-local

Although the micro accuracy of all dependencies are considerably good, the ability of current state-of-the-art statistical parsers to find difficult non-local materials is far from satisfactory, even for English (Rimell et al., 2009; Bender et al., 2011). We report the accuracy in terms of local and non-local dependencies respectively to show the difficulty of the recovery of non-local dependencies. The last four columns of Tab. 3 demonstrates the labeled/unlabeled recall of local (UR_L/LR_L) and non-local dependencies (UR_{NL}/LR_{NL}). We can clearly see that non-local dependency recovery is extremely difficult for Chinese parsing.

4.5 Deep vs. Deep

CCG and HPSG parsers also favor the dependency-based metrics for evaluation (Clark and Curran, 2007b; Miyao and Tsujii, 2008). Previous work on Chinese CCG and HPSG parsing unanimously agrees that obtaining the deep analysis of Chinese is more challenging (Yu et al., 2011; Tse and Curran, 2012). The successful C&C and Enju parsers provide very inaccurate results for Chinese texts. Though the numbers profiling the qualities of deep dependency structures under different formalisms are not directly comparable, all empirical evaluation indicates that the state-of-the-art of deep linguistic processing for Chinese lag behind very much.

5 Related Work

Wide-coverage in-depth and accurate linguistic processing is desirable for many practical NLP applications, such as machine translation (Wu et al., 2010) and information extraction (Miyao et al., 2008). Parsing in deep formalisms, e.g. CCG, HPSG, LFG and TAG, provides valuable, richer linguistic information, and researchers thus draw more and more attention to it. Very recently, study on deep linguistic processing for Chinese has been initialized. Our work is one of them.

To quickly construct deep annotations, corpus-driven grammar engineering has been studied. Phrase structure trees in CTB have been semi-automatically converted to deep derivations in the CCG (Tse and Curran, 2010), LFG (Guo et al., 2007), TAG (Xia, 2001) and HPSG (Yu et al., 2010) formalisms. Our GR extraction work is similar, but grounded in GB, which is more consistent with the construction of the original annotations.

Based on converted fine-grained linguistic annotations, successful English deep parsers, such as C&C (Clark and Curran, 2007b) and Enju (Miyao and Tsujii, 2008), have been evaluated (Yu et al., 2011; Tse and Curran, 2012). We also borrow many ideas from recent advances in deep syntactic or semantic parsing for English. In particular, Sagae and Tsujii (2008)'s and Titov et al. (2009)'s studies on transition-based deep dependency parsing motivated our work very much. However, simple adoption of their systems does not resolve Chinese GR parsing well because the GR graphs are much more complicated. Our investigation on the K-permutation transition system advances the capacity of existing methods.

6 Conclusion

Recent years witnessed rapid progress made on deep linguistic processing for English, and initial attempts for Chinese. Our work stands in between traditional dependency tree parsing and deep linguistic processing. We introduced a system for automatically extracting grammatical relations of Chinese sentences from GB phrase structure trees. The present work remedies the resource gap by facilitating the accurate extraction of GR annotations from GB trees. Manual evaluation demonstrate the effectiveness of our method. With the availability of high-quality GR resources, transition-based methods for GR parsing was studied. A new formal system, namely K-permutation system, is well theoretically discussed and practically implemented as the core module of a deep dependency parser. Empirical evaluation and analysis were presented to give better understanding of the Chinese GR parsing problem. Detailed analysis reveals some important directions for future investigation.

Acknowledgement

The work was supported by NSFC (61300064, 61170166 and 61331011) and National High-Tech R&D Program (2012AA011101).

References

Emily M. Bender, Dan Flickinger, Stephan Oepen, and Yi Zhang. 2011. Parser evaluation over local and non-local deep dependencies in a large corpus. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 397–408. Association for Computational Linguistics, Edinburgh, Scotland, UK. URL http://www.aclweb.org/anthology/D11-1037.

J. Bresnan and R. M. Kaplan. 1982. Introduction: Grammars as mental representations of language. In J. Bresnan, editor, *The Mental Representation of Grammatical Relations*, pages xvii–lii. MIT Press, Cambridge, MA.

Ted Briscoe and John Carroll. 2006. Evaluating the accuracy of an unlexicalized statistical parser on the parc depbank. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 41–48. Association for Computational Linguistics, Sydney, Australia. URL http://www.aclweb.org/anthology/P/P06/P06-2006.

Andrew Carnie. 2007. *Syntax: A Generative Introduction*. Blackwell Publishing, Blackwell Publishing 350 Main Street, Malden, MA 02148-5020, USA, second edition.

Noam Chomsky. 1981. *Lectures on Government and Binding*. Foris Publications, Dordecht.

Stephen Clark and James Curran. 2007a. Formalism-independent parser evaluation with ccg and depbank. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 248–255. Association for Computational Linguistics, Prague, Czech Republic. URL http://www.aclweb.org/anthology/P07-1032.

Stephen Clark and James R. Curran. 2007b. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Comput. Linguist.*, 33(4):493–552. URL http://dx.doi.org/10.1162/coli.2007.33.4.493.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *JOURNAL OF MACHINE LEARNING RESEARCH*, 7:551–585.

M. Dalrymple. 2001. *Lexical-Functional Grammar*, volume 34 of *Syntax and Semantics*. Academic Press, New York.

Yuqing Guo, Josef van Genabith, and Haifeng Wang. 2007. Treebank-based acquisition of lfg resources for Chinese. In *Proceedings of the LFG07 Conference*. CSLI Publications, California, USA.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the penn treebank. *Computational Linguistics*, 33(3):355–396.

Ron Kaplan, Stefan Riezler, Tracy H King, John T Maxwell III, Alex Vasserman, and Richard Crouch. 2004. Speed and accuracy in shallow and deep stochastic parsing. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 97–104. Association for Computational Linguistics, Boston, Massachusetts, USA.

Tracy Holloway King, Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ronald M. Kaplan. 2003. The PARC 700 dependency bank. In *In Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora (LINC-03)*, pages 1–8.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11. Association for Computational Linguistics, Uppsala, Sweden. URL http://www.aclweb.org/anthology/P10-1001.

Ryan McDonald. 2006. *Discriminative learning and spanning tree algorithms for dependency parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA.

Yusuke Miyao, Rune Sætre, Kenji Sagae, Takuya Matsuzaki, and Jun'ichi Tsujii. 2008. Task-oriented evaluation of syntactic parsers and their representations. In *Proceedings of ACL-08: HLT*, pages 46–54. Association for Computational Linguistics, Columbus, Ohio. URL http://www.aclweb.org/anthology/P/P08/P08-1006.

Yusuke Miyao, Kenji Sagae, and Jun'ichi Tsujii. 2007. Towards framework-independent evaluation of deep linguistic parsers. In Ann Copestake, editor, *Proceedings of the GEAF 2007 Workshop*, CSLI Studies in Computational Linguistics Online, page 21 pages. CSLI Publications. URL http://www.cs.cmu.edu/~sagae/docs/geaf07miyaoetal.pdf.

Yusuke Miyao and Jun'ichi Tsujii. 2008. Feature forest models for probabilistic hpsg parsing. *Comput. Linguist.*, 34(1):35–80. URL http://dx.doi.org/10.1162/coli.2008.34.1.35.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Comput. Linguist.*, 34:513–553. URL http://dx.doi.org/10.1162/coli.07-056-R1-07-027.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 351–359. Association for Computational Linguistics, Suntec, Singapore. URL http://www.aclweb.org/anthology/P/P09/P09-1040.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2004. Memory-based dependency parsing. In Hwee Tou Ng and Ellen Riloff, editors, *HLT-NAACL 2004 Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, pages 49–56. Association for Computational Linguistics, Boston, Massachusetts, USA.

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. The University of Chicago Press, Chicago.

Laura Rimell, Stephen Clark, and Mark Steedman. 2009. Unbounded dependency recovery for parser evaluation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 813–821. Association for Computational Linguistics, Singapore. URL http://www.aclweb.org/anthology/D/D09/D09-1085.

Kenji Sagae and Jun'ichi Tsujii. 2008. Shift-reduce dependency DAG parsing. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 753–760. Coling 2008 Organizing Committee, Manchester, UK. URL http://www.aclweb.org/anthology/C08-1095.

Weiwei Sun and Hans Uszkoreit. 2012. Capturing paradigmatic and syntagmatic lexical relations: Towards accurate Chinese part-of-speech tagging. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Weiwei Sun and Xiaojun Wan. 2013. Data-driven, pcfg-based and pseudo-pcfg-based models for Chinese dependency parsing. *Transactions of the Association for Computational Linguistics (TACL)*.

Weiwei Sun and Jia Xu. 2011. Enhancing Chinese word segmentation using unlabeled data. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 970–979. Association for Computational Linguistics, Edinburgh,

Scotland, UK. URL http://www.aclweb.org/anthology/D11-1090.

Ivan Titov, James Henderson, Paola Merlo, and Gabriele Musillo. 2009. Online graph planarisation for synchronous parsing of semantic and syntactic dependencies. In *Proceedings of the 21st international jont conference on Artifical intelligence*, pages 1562–1567. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. URL http://dl.acm.org/citation.cfm?id=1661445.1661696.

Andre Torres Martins, Noah Smith, and Eric Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 342–350. Association for Computational Linguistics, Suntec, Singapore. URL http://www.aclweb.org/anthology/P/P09/P09-1039.

Daniel Tse and James R. Curran. 2010. Chinese CCGbank: extracting CCG derivations from the penn Chinese treebank. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1083–1091. Coling 2010 Organizing Committee, Beijing, China. URL http://www.aclweb.org/anthology/C10-1122.

Daniel Tse and James R. Curran. 2012. The challenges of parsing Chinese with combinatory categorial grammar. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 295–304. Association for Computational Linguistics, Montréal, Canada. URL http://www.aclweb.org/anthology/N12-1030.

Xianchao Wu, Takuya Matsuzaki, and Jun'ichi Tsujii. 2010. Fine-grained tree-to-string translation rule extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 325–334. Association for Computational Linguistics, Uppsala, Sweden. URL http://www.aclweb.org/anthology/P10-1034.

Fei Xia. 2001. *Automatic grammar generation from two different perspectives*. Ph.D. thesis, University of Pennsylvania.

Naiwen Xue, Fei Xia, Fu-dong Chiou, and Marta Palmer. 2005. The penn Chinese treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11:207–238. URL http://portal.acm.org/citation.cfm?id=1064781.1064785.

Nianwen Xue. 2007. Tapping the implicit information for the PS to DS conversion of the Chinese treebank. In *Proceedings of the Sixth International Workshop on Treebanks and Linguistics Theories*.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *The 8th International Workshop of Parsing Technologies (IWPT2003)*, pages 195–206.

Kun Yu, Yusuke Miyao, Takuya Matsuzaki, Xiangli Wang, and Junichi Tsujii. 2011. Analysis of the difficulties in Chinese deep parsing. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 48–57. Association for Computational Linguistics, Dublin, Ireland. URL http://www.aclweb.org/anthology/W11-2907.

Kun Yu, Miyao Yusuke, Xiangli Wang, Takuya Matsuzaki, and Junichi Tsujii. 2010. Semi-automatically developing Chinese hpsg grammar from the penn Chinese treebank for deep parsing. In *Coling 2010: Posters*, pages 1417–1425. Coling 2010 Organizing Committee, Beijing, China. URL http://www.aclweb.org/anthology/C10-2162.

Yue Zhang and Stephen Clark. 2009. Transition-based parsing of the Chinese treebank using a global discriminative model. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 162–171. Association for Computational Linguistics, Paris, France. URL http://www.aclweb.org/anthology/W09-3825.

Yue Zhang and Stephen Clark. 2011. Shift-reduce CCG parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 683–692. Association for Computational Linguistics, Portland, Oregon, USA. URL http://www.aclweb.org/anthology/P11-1069.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193. Association for Computational Linguistics, Portland, Oregon, USA. URL http://www.aclweb.org/anthology/P11-2033.

Ambiguity-aware Ensemble Training for Semi-supervised Dependency Parsing

Zhenghua Li , Min Zhang*, Wenliang Chen
Provincial Key Laboratory for Computer Information Processing Technology
Soochow University
{zhli13,minzhang,wlchen}@suda.edu.cn

Abstract

This paper proposes a simple yet effective framework for semi-supervised dependency parsing at entire tree level, referred to as *ambiguity-aware ensemble training*. Instead of only using 1-best parse trees in previous work, our core idea is to utilize parse forest (*ambiguous labelings*) to combine multiple 1-best parse trees generated from diverse parsers on unlabeled data. With a conditional random field based probabilistic dependency parser, our training objective is to maximize mixed likelihood of labeled data and auto-parsed unlabeled data with ambiguous labelings. This framework offers two promising advantages. 1) ambiguity encoded in parse forests compromises noise in 1-best parse trees. During training, the parser is aware of these ambiguous structures, and has the flexibility to distribute probability mass to its preferred parse trees as long as the likelihood improves. 2) diverse syntactic structures produced by different parsers can be naturally compiled into forest, offering complementary strength to our single-view parser. Experimental results on benchmark data show that our method significantly outperforms the baseline supervised parser and other entire-tree based semi-supervised methods, such as self-training, co-training and tri-training.

1 Introduction

Supervised dependency parsing has made great progress during the past decade. However, it is very difficult to further improve performance of supervised parsers. For example, Koo and Collins (2010) and Zhang and McDonald (2012) show that incorporating higher-order features into a graph-based parser only leads to modest increase in parsing accuracy. In contrast, semi-supervised approaches, which can make use of large-scale unlabeled data, have attracted more and more interest. Previously, unlabeled data is explored to derive useful local-context features such as word clusters (Koo et al., 2008), subtree frequencies (Chen et al., 2009; Chen et al., 2013), and word co-occurrence counts (Zhou et al., 2011; Bansal and Klein, 2011). A few effective learning methods are also proposed for dependency parsing to implicitly utilize distributions on unlabeled data (Smith and Eisner, 2007; Wang et al., 2008; Suzuki et al., 2009). All above work leads to significant improvement on parsing accuracy.

Another line of research is to pick up some high-quality auto-parsed training instances from unlabeled data using bootstrapping methods, such as self-training (Yarowsky, 1995), co-training (Blum and Mitchell, 1998), and tri-training (Zhou and Li, 2005). However, these methods gain limited success in dependency parsing. Although working well on constituent parsing (McClosky et al., 2006; Huang and Harper, 2009), self-training is shown unsuccessful for dependency parsing (Spreyer and Kuhn, 2009). The reason may be that dependency parsing models are prone to amplify previous mistakes during training on self-parsed unlabeled data. Sagae and Tsujii (2007) apply a variant of co-training to dependency parsing and report positive results on out-of-domain text. Søgaard and Rishøj (2010) combine tri-training and parser ensemble to boost parsing accuracy. Both work employs two parsers to process the unlabeled data, and only select as extra training data sentences on which the 1-best parse trees of the two parsers are identical. In this way, the auto-parsed unlabeled data becomes more reliable.

*Correspondence author

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 457–467,
Baltimore, Maryland, USA, June 23-25 2014.

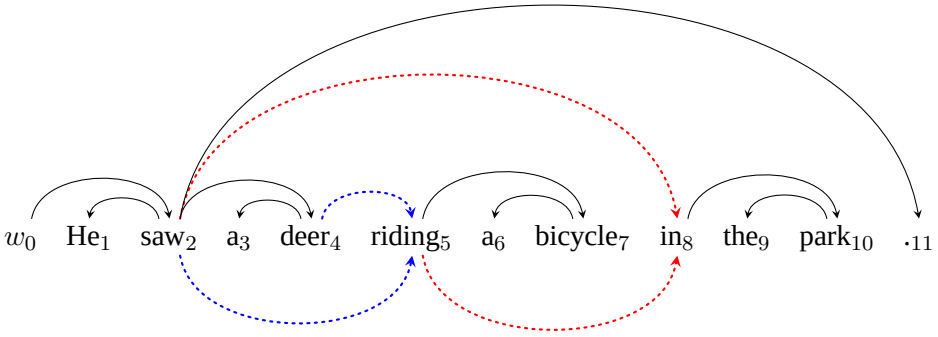


Figure 1: An example sentence with an ambiguous parse forest.

However, one obvious drawback of these methods is that they are unable to exploit unlabeled data with divergent outputs from different parsers. Our experiments show that unlabeled data with identical outputs from different parsers tends to be short (18.25 words per sentence on average), and only has a small proportion of 40% (see Table 6). More importantly, we believe that unlabeled data with divergent outputs is equally (if not more) useful. Intuitively, an unlabeled sentence with divergent outputs should contain some ambiguous syntactic structures (such as preposition phrase attachment) that are very hard to resolve and lead to the disagreement of different parsers. Such sentences can provide more discriminative instances for training which may be unavailable in labeled data.

To solve above issues, this paper proposes a more general and effective framework for semi-supervised dependency parsing, referred to as *ambiguity-aware ensemble training*. Different from traditional self/co/tri-training which only use 1-best parse trees on unlabeled data, our approach adopts ambiguous labelings, represented by parse forest, as gold-standard for unlabeled sentences. Figure 1 shows an example sentence with an ambiguous parse forest. The forest is formed by two parse trees, respectively shown at the upper and lower sides of the sentence. The differences between the two parse trees are highlighted using dashed arcs. The upper tree take *"deer"* as the subject of *"riding"*, whereas the lower one indicates that *"he"* rides the bicycle. The other difference is where the preposition phrase (PP) *"in the park"* should be attached, which is also known as the PP attachment problem, a notorious challenge for parsing. Reserving such uncertainty has three potential advantages. First, noise in unlabeled data is largely alleviated, since parse forest encodes only a few highly possible parse trees with high oracle score. Please note that the parse forest in Figure 1 contains four parse trees after combination of the two different choices. Second, the parser is able to learn useful features from the unambiguous parts of the parse forest. Finally, with sufficient unlabeled data, it is possible that the parser can learn to resolve such uncertainty by biasing to more reasonable parse trees.

To construct parse forest on unlabeled data, we employ three supervised parsers based on different paradigms, including our baseline graph-based dependency parser, a transition-based dependency parser (Zhang and Nivre, 2011), and a generative constituent parser (Petrov and Klein, 2007). The 1-best parse trees of these three parsers are aggregated in different ways. Evaluation on labeled data shows the oracle accuracy of parse forest is much higher than that of 1-best outputs of single parsers (see Table 3). Finally, using a conditional random field (CRF) based probabilistic parser, we train a better model by maximizing mixed likelihood of labeled data and auto-parsed unlabeled data with ambiguous labelings. Experimental results on both English and Chinese datasets demonstrate that the proposed ambiguity-aware ensemble training outperforms other entire-tree based methods such as self/co/tri-training. In summary, we make following contributions.

1. We propose a generalized ambiguity-aware ensemble training framework for semi-supervised dependency parsing, which can

make better use of unlabeled data, especially when parsers from different views produce divergent syntactic structures.

2. We first employ a generative constituent parser for semi-supervised dependency parsing. Experiments show that the constituent parser is very helpful since it produces more divergent structures for our semi-supervised parser than discriminative dependency parsers.

3. We build the first state-of-the-art CRF-based dependency parser. Using the probabilistic parser, we benchmark and conduct systematic comparisons among ours and all previous bootstrapping methods, including self/co/tri-training.

2 Supervised Dependency Parsing

Given an input sentence $\mathbf{x} = w_0 w_1 ... w_n$, the goal of dependency parsing is to build a dependency tree as depicted in Figure 1, denoted by $\mathbf{d} = \{(h, m) : 0 \leq h \leq n, 0 < m \leq n\}$, where (h, m) indicates a directed arc from the *head* word w_h to the *modifier* w_m, and w_0 is an artificial node linking to the root of the sentence.

In parsing community, two mainstream methods tackle the dependency parsing problem from different perspectives but achieve comparable accuracy on a variety of languages. The graph-based method views the problem as finding an optimal tree from a fully-connected directed graph (McDonald et al., 2005; McDonald and Pereira, 2006; Carreras, 2007; Koo and Collins, 2010), while the transition-based method tries to find a highest-scoring transition sequence that leads to a legal dependency tree (Yamada and Matsumoto, 2003; Nivre, 2003; Zhang and Nivre, 2011).

2.1 Graph-based Dependency Parser (GParser)

In this work, we adopt the graph-based paradigm because it allows us to naturally derive conditional probability of a dependency tree $\mathbf{d}$ given a sentence $\mathbf{x}$, which is required to compute likelihood of both labeled and unlabeled data. Under the graph-based model, the score of a dependency tree is factored into the scores of small subtrees $\mathbf{p}$.

$$\begin{aligned} Score(\mathbf{x}, \mathbf{d}; \mathbf{w}) &= \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{d}) \\ &= \sum_{\mathbf{p} \subseteq \mathbf{d}} Score(\mathbf{x}, \mathbf{p}; \mathbf{w}) \end{aligned}$$

(a) *single dependency* (b) *adjacent sibling*

Figure 2: Two types of scoring subtrees in our second-order graph-based parsers.

Dependency features $\mathbf{f}_{dep}(\mathbf{x}, h, m)$:
w_h, w_m, t_h, t_m, $t_{h\pm1}$, $t_{m\pm1}$, t_b, $dir(h,m)$, $dist(h,m)$
Sibling features $\mathbf{f}_{sib}(\mathbf{x}, h, m, s)$:
w_h, w_s, w_m, t_h, t_m, t_s, $t_{h\pm1}$, $t_{m\pm1}$, $t_{s\pm1}$
$dir(h,m)$, $dist(h,m)$

Table 1: Brief illustration of the syntactic features. t_i denotes the POS tag of w_i. b is an index between h and m. $dir(i,j)$ and $dist(i,j)$ denote the direction and distance of the dependency (i,j).

We adopt the second-order graph-based dependency parsing model of McDonald and Pereira (2006) as our core parser, which incorporates features from the two kinds of subtrees in Fig. 2.[1] Then the score of a dependency tree is:

$$\begin{aligned} Score(\mathbf{x}, \mathbf{d}; \mathbf{w}) = &\sum_{\{(h,m)\} \subseteq \mathbf{d}} \mathbf{w}_{dep} \cdot \mathbf{f}_{dep}(\mathbf{x}, h, m) \\ &+ \sum_{\{(h,s),(h,m)\} \subseteq \mathbf{d}} \mathbf{w}_{sib} \cdot \mathbf{f}_{sib}(\mathbf{x}, h, s, m) \end{aligned}$$

where $\mathbf{f}_{dep}(\mathbf{x}, h, m)$ and $\mathbf{f}_{sib}(\mathbf{x}, h, s, m)$ are the feature vectors of the two subtree in Fig. 2; $\mathbf{w}_{dep/sib}$ are feature weight vectors; the dot product gives scores contributed by corresponding subtrees.

For syntactic features, we adopt those of Bohnet (2010) which include two categories corresponding to the two types of scoring subtrees in Fig. 2. We summarize the atomic features used in each feature category in Table 1. These atomic features are concatenated in different combinations to compose rich feature sets. Please refer to Table 4 of Bohnet (2010) for the complete feature list.

2.2 CRF-based GParser

Previous work on graph-based dependency parsing mostly adopts linear models and perceptron based training procedures, which lack probabilistic explanations of dependency trees and do not need to compute likelihood of labeled training

[1] Higher-order models of Carreras (2007) and Koo and Collins (2010) can achieve higher accuracy, but has much higher time cost ($O(n^4)$). Our approach is applicable to these higher-order models, which we leave for future work.

data. Instead, we build a log-linear CRF-based dependency parser, which is similar to the CRF-based constituent parser of Finkel et al. (2008). Assuming the feature weights $\mathbf{w}$ are known, the probability of a dependency tree $\mathbf{d}$ given an input sentence $\mathbf{x}$ is defined as:

$$p(\mathbf{d}|\mathbf{x};\mathbf{w}) = \frac{exp\{Score(\mathbf{x},\mathbf{d};\mathbf{w})\}}{Z(\mathbf{x};\mathbf{w})}$$
$$Z(\mathbf{x};\mathbf{w}) = \sum_{\mathbf{d}' \in \mathcal{Y}(\mathbf{x})} exp\{Score(\mathbf{x},\mathbf{d}';\mathbf{w})\} \quad (1)$$

where $Z(\mathbf{x})$ is the normalization factor and $\mathcal{Y}(\mathbf{x})$ is the set of all legal dependency trees for $\mathbf{x}$.

Suppose the labeled training data is $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{d}_i)\}_{i=1}^{N}$. Then the log likelihood of $\mathcal{D}$ is:

$$\mathcal{L}(\mathcal{D};\mathbf{w}) = \sum_{i=1}^{N} \log p(\mathbf{d}_i|\mathbf{x}_i;\mathbf{w})$$

The training objective is to maximize the log likelihood of the training data $\mathcal{L}(\mathcal{D})$. The partial derivative with respect to the feature weights $\mathbf{w}$ is:

$$\frac{\partial \mathcal{L}(\mathcal{D};\mathbf{w})}{\partial \mathbf{w}} = \sum_{i=1}^{N} \left(\mathbf{f}(\mathbf{x}_i,\mathbf{d}_i) - \sum_{\mathbf{d}' \in \mathcal{Y}(\mathbf{x}_i)} p(\mathbf{d}'|\mathbf{x}_i;\mathbf{w}) \mathbf{f}(\mathbf{x}_i,\mathbf{d}') \right) \quad (2)$$

where the first term is the empirical counts and the second term is the model expectations. Since $\mathcal{Y}(\mathbf{x}_i)$ contains exponentially many dependency trees, direct calculation of the second term is prohibitive. Instead, we can use the classic inside-outside algorithm to efficiently compute the model expectations within $O(n^3)$ time complexity, where n is the input sentence length.

3 Ambiguity-aware Ensemble Training

In standard entire-tree based semi-supervised methods such as self/co/tri-training, automatically parsed unlabeled sentences are used as additional training data, and noisy 1-best parse trees are considered as gold-standard. To alleviate the noise, the tri-training method only uses unlabeled data on which multiple parsers from different views produce identical parse trees. However, unlabeled data with divergent syntactic structures should be more useful. Intuitively, if several parsers disagree on an unlabeled sentence, it implies that the unlabeled sentence contains some difficult syntactic phenomena which are not sufficiently covered in manually labeled data. Therefore, exploiting such unlabeled data may introduce more discriminative syntactic knowledge, largely compensating labeled training data.

To address above issues, we propose *ambiguity-aware ensemble training*, which can be interpreted as a *generalized tri-training* framework. The key idea is the use of *ambiguous labelings* for the purpose of aggregating multiple 1-best parse trees produced by several diverse parsers. Here, "ambiguous labelings" mean an unlabeled sentence may have multiple parse trees as gold-standard reference, represented by parse forest (see Figure 1). The training procedure aims to maximize mixed likelihood of both manually labeled and auto-parsed unlabeled data with ambiguous labelings. For an unlabeled instance, the model is updated to maximize the probability of its parse forest, instead of a single parse tree in traditional tri-training. In other words, the model is free to distribute probability mass among the trees in the parse forest to its liking, as long as the likelihood improves (Täckström et al., 2013).

3.1 Likelihood of the Unlabeled Data

The auto-parsed unlabeled data with ambiguous labelings is denoted as $\mathcal{D}' = \{(\mathbf{u}_i, \mathcal{V}_i)\}_{i=1}^{M}$, where $\mathbf{u}_i$ is an unlabeled sentence, and $\mathcal{V}_i$ is the corresponding parse forest. Then the log likelihood of $\mathcal{D}'$ is:

$$\mathcal{L}(\mathcal{D}';\mathbf{w}) = \sum_{i=1}^{M} \log \left(\sum_{\mathbf{d}' \in \mathcal{V}_i} p(\mathbf{d}'|\mathbf{u}_i;\mathbf{w}) \right)$$

where $p(\mathbf{d}'|\mathbf{u}_i;\mathbf{w})$ is the conditional probability of $\mathbf{d}'$ given $\mathbf{u}_i$, as defined in Eq. (1). For an unlabeled sentence $\mathbf{u}_i$, the probability of its parse forest $\mathcal{V}_i$ is the summation of the probabilities of all the parse trees contained in the forest.

Then we can derive the partial derivative of the log likelihood with respect to $\mathbf{w}$:

$$\frac{\partial \mathcal{L}(\mathcal{D}';\mathbf{w})}{\partial \mathbf{w}} = \sum_{i=1}^{M} \left(\sum_{\mathbf{d}' \in \mathcal{V}_i} \tilde{p}(\mathbf{d}'|\mathbf{u}_i,\mathcal{V}_i;\mathbf{w}) \mathbf{f}(\mathbf{u}_i,\mathbf{d}') - \sum_{\mathbf{d}' \in \mathcal{Y}(\mathbf{u}_i)} p(\mathbf{d}'|\mathbf{u}_i;\mathbf{w}) \mathbf{f}(\mathbf{u}_i,\mathbf{d}') \right) \quad (3)$$

where $\tilde{p}(\mathbf{d}'|\mathbf{u}_i,\mathcal{V}_i;\mathbf{w})$ is the probability of $\mathbf{d}'$ un-

der the space constrained by the parse forest $\mathcal{V}_i$.

$$\tilde{p}(\mathbf{d}'|\mathbf{u}_i, \mathcal{V}_i; \mathbf{w}) = \frac{\exp\{Score(\mathbf{u}_i, \mathbf{d}'; \mathbf{w})\}}{Z(\mathbf{u}_i, \mathcal{V}_i; \mathbf{w})}$$

$$Z(\mathbf{u}_i, \mathcal{V}_i; \mathbf{w}) = \sum_{\mathbf{d}' \in \mathcal{V}_i} \exp\{Score(\mathbf{u}_i, \mathbf{d}'; \mathbf{w})\}$$

The second term in Eq. (3) is the same with the second term in Eq. (2). The first term in Eq. (3) can be efficiently computed by running the inside-outside algorithm in the constrained search space $\mathcal{V}_i$.

3.2 Stochastic Gradient Descent (SGD) Training

We apply L2-norm regularized SGD training to iteratively learn feature weights $\mathbf{w}$ for our CRF-based baseline and semi-supervised parsers. We follow the implementation in CRFsuite.[2] At each step, the algorithm approximates a gradient with a small subset of the training examples, and then updates the feature weights. Finkel et al. (2008) show that SGD achieves optimal test performance with far fewer iterations than other optimization routines such as L-BFGS. Moreover, it is very convenient to parallel SGD since computations among examples in the same batch is mutually independent.

Training with the combined labeled and unlabeled data, the objective is to maximize the mixed likelihood:

$$\mathcal{L}(\mathcal{D}; \mathcal{D}') = \mathcal{L}(\mathcal{D}) + \mathcal{L}(\mathcal{D}')$$

Since $\mathcal{D}'$ contains much more instances than $\mathcal{D}$ (1.7M vs. 40K for English, and 4M vs. 16K for Chinese), it is likely that the unlabeled data may overwhelm the labeled data during SGD training. Therefore, we propose a simple corpus-weighting strategy, as shown in Algorithm 1, where $\mathcal{D}^b_{i,k}$ is the subset of training data used in k^{th} update and b is the batch size; η_k is the update step, which is adjusted following the simulated annealing procedure (Finkel et al., 2008). The idea is to use a fraction of training data ($\mathcal{D}_i$) at each iteration, and do corpus weighting by randomly sampling labeled and unlabeled instances in a certain proportion (N_1 vs. M_1).

Once the feature weights $\mathbf{w}$ are learnt, we can parse the test data to find the optimal parse tree.

$$\mathbf{d}^* = \arg\max_{\mathbf{d}' \in \mathcal{Y}(\mathbf{x})} p(\mathbf{d}'|\mathbf{x}; \mathbf{w}) = \arg\max_{\mathbf{d}' \in \mathcal{Y}(\mathbf{x})} Score(\mathbf{x}, \mathbf{d}'; \mathbf{w})$$

This can be done with the Viterbi decoding algorithm described in McDonald and Pereira (2006) in $O(n^3)$ parsing time.

Algorithm 1 SGD training with mixed labeled and unlabeled data.

Input: Labeled data $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{d}_i)\}_{i=1}^N$, and unlabeled data $\mathcal{D}' = \{(\mathbf{u}_i, \mathcal{V}_i)\}_{j=1}^M$; Parameters: I, N_1, M_1, b
Output: $\mathbf{w}$
Initialization: $\mathbf{w}^{(0)} = \mathbf{0}$, $k = 0$;
for $i = 1$ **to** I **do** {*iterations*}
  Randomly select N_1 instances from $\mathcal{D}$ and M_1 instances from $\mathcal{D}'$ to compose a new dataset $\mathcal{D}_i$, and shuffle it.
  Traverse $\mathcal{D}_i$: a small batch $\mathcal{D}^b_{i,k} \subseteq \mathcal{D}_i$ at one step.
    $\mathbf{w}_{k+1} = \mathbf{w}_k + \eta_k \frac{1}{b} \nabla \mathcal{L}(\mathcal{D}^b_{i,k}; \mathbf{w}_k)$
    $k = k + 1$
end for

3.3 Forest Construction with Diverse Parsers

To construct parse forests for unlabeled data, we employ three diverse parsers, i.e., our baseline GParser, a transition-based parser (ZPar[3]) (Zhang and Nivre, 2011), and a generative constituent parser (Berkeley Parser[4]) (Petrov and Klein, 2007). These three parsers are trained on labeled data and then used to parse each unlabeled sentence. We aggregate the three parsers' outputs on unlabeled data in different ways and evaluate the effectiveness through experiments.

4 Experiments and Analysis

To verify the effectiveness of our proposed approach, we conduct experiments on Penn Treebank (PTB) and Penn Chinese Treebank 5.1 (CTB5). For English, we follow the popular practice to split data into training (sections 2-21), development (section 22), and test (section 23). For CTB5, we adopt the data split of (Duan et al., 2007). We convert original bracketed structures into dependency structures using Penn2Malt with its default head-finding rules.

For unlabeled data, we follow Chen et al. (2013) and use the BLLIP WSJ corpus (Charniak et al., 2000) for English and Xinhua portion of Chinese

[2] http://www.chokkan.org/software/crfsuite/

[3] http://people.sutd.edu.sg/~yue_zhang/doc/

[4] https://code.google.com/p/berkeleyparser/

	Train	Dev	Test	Unlabeled
PTB	39,832	1,700	2,416	1.7M
CTB5	16,091	803	1,910	4M

Table 2: Data sets (in sentence number).

Gigaword Version 2.0 (LDC2009T14) (Huang, 2009) for Chinese. We build a CRF-based bigram part-of-speech (POS) tagger with the features described in (Li et al., 2012), and produce POS tags for all train/development/test/unlabeled sets (10-way jackknifing for training sets). The tagging accuracy on test sets is 97.3% on English and 94.0% on Chinese. Table 2 shows the data statistics.

We measure parsing performance using the standard unlabeled attachment score (UAS), excluding punctuation marks. For significance test, we adopt Dan Bikel's randomized parsing evaluation comparator (Noreen, 1989).[5]

4.1 Parameter Setting

When training our CRF-based parsers with SGD, we use the batch size $b = 100$ for all experiments. We run SGD for $I = 100$ iterations and choose the model that performs best on development data. For the semi-supervised parsers trained with Algorithm 1, we use $N_1 = 20$K and $M_1 = 50$K for English, and $N_1 = 15$K and $M_1 = 50$K for Chinese, based on a few preliminary experiments. To accelerate the training, we adopt parallelized implementation of SGD and employ 20 threads for each run. For semi-supervised cases, one iteration takes about 2 hours on an IBM server having 2.0 GHz Intel Xeon CPUs and 72G memory.

Default parameter settings are used for training ZPar and Berkeley Parser. We run ZPar for 50 iterations, and choose the model that achieves highest accuracy on the development data. For Berkeley Parser, we use the model after 5 split-merge iterations to avoid over-fitting the training data according to the manual. The phrase-structure outputs of Berkeley Parser are converted into dependency structures using the same head-finding rules.

4.2 Methodology Study on Development Data

Using three supervised parsers, we have many options to construct parse forest on unlabeled data. To examine the effect of different ways for forest construction, we conduct extensive methodology study on development data. Table 3 presents the results. We divide the systems into three types: 1) supervised single parsers; 2) CRF-based GParser with conventional self/co/tri-training; 3) CRF-based GParser with our approach. For the latter two cases, we also present the oracle accuracy and averaged head number per word ("Head/Word") of parse forest when applying different ways to construct forests on development datasets.

The first major row presents performance of the three supervised parsers. We can see that the three parsers achieve comparable performance on English, but the performance of ZPar is largely inferior on Chinese.

The second major row shows the results when we use single 1-best parse trees on unlabeled data. When using the outputs of GParser itself ("Unlabeled ← G"), the experiment reproduces traditional self-training. The results on both English and Chinese re-confirm that *self-training may not work for dependency parsing*, which is consistent with previous studies (Spreyer and Kuhn, 2009). The reason may be that dependency parsers are prone to amplify previous mistakes on unlabeled data during training.

The next two experiments in the second major row reimplement *co-training*, where another parser's 1-best results are projected into unlabeled data to help the core parser. Using unlabeled data with the results of ZPar ("Unlabeled ← Z") significantly outperforms the baseline GParser by 0.30% (93.15-82.85) on English. However, the improvement on Chinese is not significant. Using unlabeled data with the results of Berkeley Parser ("Unlabeled ← B") significantly improves parsing accuracy by 0.55% (93.40-92.85) on English and 1.06% (83.34-82.28) on Chinese. We believe the reason is that being a generative model designed for constituent parsing, Berkeley Parser is more different from discriminative dependency parsers, and therefore can provide more divergent syntactic structures. This kind of syntactic divergence is helpful because it can provide complementary knowledge from a different perspective. Surdeanu and Manning (2010) also show that the diversity of parsers is important for performance improvement when integrating different parsers in the supervised track. Therefore, we can conclude that *co-training helps dependency parsing, especially when using a more divergent parser*.

The last experiment in the second major row is known as *tri-training*, which only uses unla-

[5] http://www.cis.upenn.edu/~dbikel/software.html

		English			Chinese		
		UAS	Oracle	Head/Word	UAS	Oracle	Head/Word
Supervised	GParser	**92.85**	—	—	82.28	—	—
	ZPar	92.50			81.04		
	Berkeley	92.70			**82.46**		
Semi-supervised GParser with Single 1-best Trees	Unlabeled ← G (self-train)	92.88	92.85	1.000	82.14	82.28	1.000
	Unlabeled ← Z (co-train)	93.15 †	92.50		82.54	81.04	
	Unlabeled ← B (co-train)	93.40 †	92.70		**83.34** †	82.46	
	Unlabeled ← B=Z (tri-train)	**93.50** †	97.52		83.10 †	95.05	
Semi-supervised GParser Ambiguity-aware Ensemble	Unlabeled ← Z+G	93.18 †	94.97	1.053	82.78	86.66	1.136
	Unlabeled ← B+G	93.35 †	96.37	1.080	83.24 †	89.72	1.188
	Unlabeled ← B+Z	**93.78** †‡	96.18	1.082	83.86 †‡	89.54	1.199
	Unlabeled ← B+(Z∩G)	93.77 †‡	95.60	1.050	**84.26** †‡	87.76	1.106
	Unlabeled ← B+Z+G	93.50 †	96.95	1.112	83.30 †	91.50	1.281

Table 3: Main results on development data. G is short for GParser, Z for ZPar, and B for Berkeley Parser. † means the corresponding parser significantly outperforms supervised parsers, and ‡ means the result significantly outperforms co/tri-training at confidence level of $p < 0.01$.

beled sentences on which Berkeley Parser and ZPar produce identical outputs ("Unlabeled ← B=Z"). We can see that with the verification of two views, the oracle accuracy is much higher than using single parsers (97.52% vs. 92.85% on English, and 95.06% vs. 82.46% on Chinese). Although using less unlabeled sentences (0.7M for English and 1.2M for Chinese), *tri-training achieves comparable performance to co-training* (slightly better on English and slightly worse on Chinese).

The third major row shows the results of the semi-supervised GParser with our proposed approach. We experiment with different combinations of the 1-best parse trees of the three supervised parsers. The first three experiments combine 1-best outputs of two parsers to compose parse forest on unlabeled data. "Unlabeled ← B+(Z∩G)" means that the parse forest is initialized with the Berkeley parse and augmented with the intersection of dependencies of the 1-best outputs of ZPar and GParser. In the last setting, the parse forest contains all three 1-best results.

When the parse forests of the unlabeled data are the union of the outputs of GParser and ZPar, denoted as "Unlabeled ← Z+G", each word has 1.053 candidate heads on English and 1.136 on Chinese, and the oracle accuracy is higher than using 1-best outputs of single parsers (94.97% vs. 92.85% on English, 86.66% vs. 82.46% on Chinese). However, we find that although the parser significantly outperforms the supervised GParser on English, it does not gain significant improvement over co-training with ZPar ("Unlabeled ← Z") on both English and Chinese.

Combining the outputs of Berkeley Parser and GParser ("Unlabeled ← B+G"), we get higher oracle score (96.37% on English and 89.72% on Chinese) and higher syntactic divergence (1.085 candidate heads per word on English, and 1.188 on Chinese) than "Unlabeled ← Z+G", which verifies our earlier discussion that Berkeley Parser produces more different structures than ZPar. However, it leads to slightly worse accuracy than co-training with Berkeley Parser ("Unlabeled ← B"). This indicates that adding the outputs of GParser itself does not help the model.

Combining the outputs of Berkeley Parser and ZPar ("Unlabeled ← B+Z"), we get the best performance on English, which is also significantly better than both co-training ("Unlabeled ← B") and tri-training ("Unlabeled ← B=Z") on both English and Chinese. This demonstrates that *our proposed approach can better exploit unlabeled data than traditional self/co/tri-training*. More analysis and discussions are in Section 4.4.

During experimental trials, we find that "Unlabeled ← B+(Z∩G)" can further boost performance on Chinese. A possible explanation is that by using the intersection of the outputs of GParser and ZPar, the size of the parse forest is better controlled, which is helpful considering that ZPar performs worse on this data than both Berkeley Parser and GParser.

Adding the output of GParser itself ("Unlabeled ← B+Z+G") leads to accuracy drop, although the oracle score is higher (96.95% on English and 91.50% on Chinese) than "Unlabeled ← B+Z". We suspect the reason is that the model is likely to distribute the probability mass to these parse trees produced by itself instead of those by Berkeley Parser or ZPar under this setting.

	Sup	Semi
McDonald and Pereira (2006)	91.5	—
Koo and Collins (2010) [higher-order]	93.04	
Zhang and McDonald (2012) [higher-order]	93.06	
Zhang and Nivre (2011) [higher-order]	92.9	
Koo et al. (2008) [higher-order]	92.02	93.16
Chen et al. (2009) [higher-order]	92.40	93.16
Suzuki et al. (2009) [higher-order,cluster]	92.70	93.79
Zhou et al. (2011) [higher-order]	91.98	92.64
Chen et al. (2013) [higher-order]	92.76	93.77
This work	92.34	93.19

Table 4: UAS comparison on English test data.

In summary, we can conclude that *our proposed ambiguity-aware ensemble training is significantly better than both the supervised approaches and the semi-supervised approaches that use 1-best parse trees*. Appropriately composing the forest parse, our approach outperforms the best results of co-training or tri-training by 0.28% (93.78-93.50) on English and 0.92% (84.26-83.34) on Chinese.

4.3 Comparison with Previous Work

We adopt the best settings on development data for semi-supervised GParser with our proposed approach, and make comparison with previous results on test data. Table 4 shows the results.

The first major row lists several state-of-the-art supervised methods. McDonald and Pereira (2006) propose a second-order graph-based parser, but use a smaller feature set than our work. Koo and Collins (2010) propose a third-order graph-based parser. Zhang and McDonald (2012) explore higher-order features for graph-based dependency parsing, and adopt beam search for fast decoding. Zhang and Nivre (2011) propose a feature-rich transition-based parser. All work in the second major row adopts semi-supervised methods. The results show that our approach achieves comparable accuracy with most previous semi-supervised methods. Both Suzuki et al. (2009) and Chen et al. (2013) adopt the higher-order parsing model of Carreras (2007), and Suzuki et al. (2009) also incorporate word cluster features proposed by Koo et al. (2008) in their system. We expect our approach may achieve higher performance with such enhancements, which we leave for future work. Moreover, our method may be combined with other semi-supervised approaches, since they are orthogonal in methodology and utilize unlabeled data from different perspectives.

Table 5 make comparisons with previous results on Chinese test data. Li et al. (2012) and Bohnet and Nivre (2012) use joint models for POS tagging and dependency parsing, significantly outperforming their pipeline counterparts. Our approach can be combined with their work to utilize unlabeled data to improve both POS tagging and parsing simultaneously. Our work achieves comparable accuracy with Chen et al. (2013), although they adopt the higher-order model of Carreras (2007). Again, our method may be combined with their work to achieve higher performance.

		UAS
Supervised	Li et al. (2012) [joint]	82.37
	Bohnet and Nivre (2012) [joint]	81.42
	Chen et al. (2013) [higher-order]	81.01
	This work	81.14
Semi	Chen et al. (2013) [higher-order]	83.08
	This work	82.89

Table 5: UAS comparison on Chinese test data.

Unlabeled data	UAS	#Sent	Len	Head/Word	Oracle
NULL	92.34	0	—	—	—
Consistent (tri-train)	92.94	0.7M	18.25	1.000	97.65
Low divergence	92.94	0.5M	28.19	1.062	96.53
High divergence	93.03	0.5M	27.85	1.211	94.28
ALL	93.19	1.7M	24.15	1.087	96.09

Table 6: Performance of our semi-supervised GParser with different sets of "Unlabeled ← B+Z" on English test set. "Len" means averaged sentence length.

4.4 Analysis

To better understand the effectiveness of our proposed approach, we make detailed analysis using the semi-supervised GParser with "Unlabeled ← B+Z" on English datasets.

Contribution of unlabeled data with regard to syntactic divergence: We divide the unlabeled data into three sets according to the divergence of the 1-best outputs of Berkeley Parser and ZPar. The first set contains those sentences that the two parsers produce identical parse trees, denoted by "consistent", which corresponds to the setting for tri-training. Other sentences are split into two sets according to averaged number of heads per word in parse forests, denoted by "low divergence" and "high divergence" respectively. Then we train semi-supervised GParser using the three sets of unlabeled data. Table 6 illustrates the results and statistics. We can see that unlabeled data with identical outputs from Berkeley Parser and ZPar tends to be short sentences (18.25 words per sen-

tence on average). Results show all the three sets of unlabeled data can help the parser. Especially, the unlabeled data with highly divergent structures leads to slightly higher improvement. This demonstrates that *our approach can better exploit unlabeled data on which parsers of different views produce divergent structures*.

Impact of unlabeled data size: To understand how our approach performs with regards to the unlabeled data size, we train semi-supervised GParser with different sizes of unlabeled data. Fig. 3 shows the accuracy curve on the test set. We can see that the parser consistently achieves higher accuracy with more unlabeled data, demonstrating the effectiveness of our approach. We expect that our approach has potential to achieve higher accuracy with more additional data.

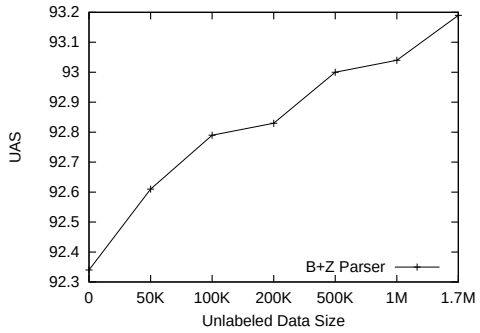


Figure 3: Performance of GParser with different sizes of "Unlabeled ← B+Z" on English test set.

5 Related Work

Our work is originally inspired by the work of Täckström et al. (2013). They first apply the idea of ambiguous labelings to multilingual parser transfer in the unsupervised parsing field, which aims to build a dependency parser for a resource-poor target language by making use of source-language treebanks. Different from their work, we explore the idea for semi-supervised dependency parsing where a certain amount of labeled training data is available. Moreover, we for the first time build a state-of-the-art CRF-based dependency parser and conduct in-depth comparisons with previous methods. Similar ideas of learning with ambiguous labelings are previously explored for classification (Jin and Ghahramani, 2002) and sequence labeling problems (Dredze et al., 2009).

Our work is also related with the parser ensemble approaches such as stacked learning and re-parsing in the supervised track. Stacked learning uses one parser's outputs as guide features for another parser, leading to improved performance (Nivre and McDonald, 2008; Torres Martins et al., 2008). Re-parsing merges the outputs of several parsers into a dependency graph, and then apply Viterbi decoding to find a better tree (Sagae and Lavie, 2006; Surdeanu and Manning, 2010). One possible drawback of parser ensemble is that several parsers are required to parse the same sentence during the test phase. Moreover, our approach can benefit from these methods in that we can get parse forests of higher quality on unlabeled data (Zhou, 2009).

6 Conclusions

This paper proposes a generalized training framework of semi-supervised dependency parsing based on ambiguous labelings. For each unlabeled sentence, we combine the 1-best parse trees of several diverse parsers to compose ambiguous labelings, represented by a parse forest. The training objective is to maximize the mixed likelihood of both the labeled data and the auto-parsed unlabeled data with ambiguous labelings. Experiments show that our framework can make better use of the unlabeled data, especially those with divergent outputs from different parsers, than traditional tri-training. Detailed analysis demonstrates the effectiveness of our approach. Specifically, we find that our approach is very effective when using divergent parsers such as the generative parser, and it is also helpful to properly balance the size and oracle accuracy of the parse forest of the unlabeled data.

For future work, among other possible extensions, we would like to see how our approach performs when employing more diverse parsers to compose the parse forest of higher quality for the unlabeled data, such as the easy-first non-directional dependency parser (Goldberg and Elhadad, 2010) and other constituent parsers (Collins and Koo, 2005; Charniak and Johnson, 2005; Finkel et al., 2008).

Acknowledgments

The authors would like to thank the critical and insightful comments from our anonymous reviewers. This work was supported by National Natural Science Foundation of China (Grant No. 61373095, 61333018).

References

Mohit Bansal and Dan Klein. 2011. Web-scale features for full-scale parsing. In *Proceedings of ACL*, pages 693–702.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 92–100.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of EMNLP 2012*, pages 1455–1465.

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of COLING*, pages 89–97.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of EMNLP/CoNLL*, pages 141–150.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of ACL*, pages 173–180.

Eugene Charniak, Don Blaheta, Niyu Ge, Keith Hall, John Hale, and Mark Johnson. 2000. BLLIP 1987-89 WSJ Corpus Release 1, LDC2000T43. *Linguistic Data Consortium*.

Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009. Improving dependency parsing with subtrees from auto-parsed data. In *Proceedings of EMNLP*, pages 570–579.

Wenliang Chen, Min Zhang, and Yue Zhang. 2013. Semi-supervised feature transformation for dependency parsing. In *Proceedings of EMNLP*, pages 1303–1313.

Michael J. Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, pages 25–70.

Mark Dredze, Partha Pratim Talukdar, and Koby Crammer. 2009. Sequence learning from data with multiple labels. In *ECML/PKDD Workshop on Learning from Multi-Label Data*.

Xiangyu Duan, Jun Zhao, and Bo Xu. 2007. Probabilistic models for action-based Chinese dependency parsing. In *Proceedings of ECML/ECPPKDD*, pages 559–566.

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *Proceedings of ACL*, pages 959–967.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Proceedings of NAACL*.

Zhongqiang Huang and Mary Harper. 2009. Self-training PCFG grammars with latent annotations across languages. In *Proceedings of EMNLP 2009*, pages 832–841.

Chu-Ren Huang. 2009. Tagged Chinese Gigaword Version 2.0, LDC2009T14. *Linguistic Data Consortium*.

Rong Jin and Zoubin Ghahramani. 2002. Learning with multiple labels. In *Proceedings of NIPS*.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *ACL*, pages 1–11.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL*, pages 595–603.

Zhenghua Li, Min Zhang, Wanxiang Che, and Ting Liu. 2012. A separately passive-aggressive training algorithm for joint POS tagging and dependency parsing. In *COLING 2012*, pages 1681–1698.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL*, pages 152–159.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, pages 81–88.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL*, pages 91–98.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL*, pages 950–958.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of IWPT*, pages 149–160.

Eric W. Noreen. 1989. *Computer-intensive methods for testing hypotheses: An introduction*. John Wiley & Sons, Inc., New York.

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

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of NAACL*, pages 129–132.

Kenji Sagae and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, pages 1044–1050.

David A. Smith and Jason Eisner. 2007. Bootstrapping feature-rich dependency parsers with entropic priors. In *Proceedings of EMNLP*, pages 667–677.

Anders Søgaard and Christian Rishøj. 2010. Semi-supervised dependency parsing using generalized tri-training. In *Proceedings of ACL*, pages 1065–1073.

Kathrin Spreyer and Jonas Kuhn. 2009. Data-driven dependency parsing of new languages using incomplete and noisy training data. In *CoNLL*, pages 12–20.

Mihai Surdeanu and Christopher D. Manning. 2010. Ensemble models for dependency parsing: Cheap and good? In *Proceedings of NAACL*, pages 649–652.

Jun Suzuki, Hideki Isozaki, Xavier Carreras, and Michael Collins. 2009. An empirical study of semi-supervised structured conditional models for dependency parsing. In *Proceedings of EMNLP*, pages 551–560.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *Proceedings of NAACL*, pages 1061–1071.

André Filipe Torres Martins, Dipanjan Das, Noah A. Smith, and Eric P. Xing. 2008. Stacking dependency parsers. In *Proceedings of EMNLP*, pages 157–166.

Qin Iris Wang, Dale Schuurmans, and Dekang Lin. 2008. Semi-supervised convex training for dependency parsing. In *Proceedings of ACL*, pages 532–540.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, pages 195–206.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of ACL*, pages 189–196.

Hao Zhang and Ryan McDonald. 2012. Generalized higher-order dependency parsing with cube pruning. In *Proceedings of EMNLP-CoNLL*, pages 320–331.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of ACL*, pages 188–193.

Zhi-Hua Zhou and Ming Li. 2005. Tri-training: Exploiting unlabeled data using three classifiers. In *IEEE Transactions on Knowledge and Data Engineering*, pages 1529–1541.

Guangyou Zhou, Jun Zhao, Kang Liu, and Li Cai. 2011. Exploiting web-derived selectional preference to improve statistical dependency parsing. In *Proceedings of ACL*, pages 1556–1565.

Zhi-Hua Zhou. 2009. When semi-supervised learning meets ensemble learning. In *MCS*.

A Robust Approach to Aligning Heterogeneous Lexical Resources

Mohammad Taher Pilehvar and **Roberto Navigli**
Department of Computer Science
Sapienza University of Rome
{pilehvar,navigli}@di.uniroma1.it

Abstract

Lexical resource alignment has been an active field of research over the last decade. However, prior methods for aligning lexical resources have been either specific to a particular pair of resources, or heavily dependent on the availability of hand-crafted alignment data for the pair of resources to be aligned. Here we present a unified approach that can be applied to an arbitrary pair of lexical resources, including machine-readable dictionaries with no network structure. Our approach leverages a similarity measure that enables the structural comparison of senses across lexical resources, achieving state-of-the-art performance on the task of aligning WordNet to three different collaborative resources: Wikipedia, Wiktionary and OmegaWiki.

1 Introduction

Lexical resources are repositories of machine-readable knowledge that can be used in virtually any Natural Language Processing task. Notable examples are WordNet, Wikipedia and, more recently, collaboratively-curated resources such as OmegaWiki and Wiktionary (Hovy et al., 2013). On the one hand, these resources are heterogeneous in design, structure and content, but, on the other hand, they often provide complementary knowledge which we would like to see integrated. Given the large scale this intrinsic issue can only be addressed automatically, by means of lexical resource alignment algorithms. Owing to its ability to bring together features like multilinguality and increasing coverage, over the past few years resource alignment has proven beneficial to a wide spectrum of tasks, such as Semantic Parsing (Shi and Mihalcea, 2005), Semantic Role Labeling (Palmer et al., 2010), and Word Sense Disambiguation (Navigli and Ponzetto, 2012).

Nevertheless, when it comes to aligning textual definitions in different resources, the lexical approach (Ruiz-Casado et al., 2005; de Melo and Weikum, 2010; Henrich et al., 2011) falls short because of the potential use of totally different wordings to define the same concept. Deeper approaches leverage semantic similarity to go beyond the surface realization of definitions (Navigli, 2006; Meyer and Gurevych, 2011; Niemann and Gurevych, 2011). While providing good results in general, these approaches fail when the definitions of a given word are not of adequate quality and expressiveness to be distinguishable from one another. When a lexical resource can be viewed as a semantic graph, as with WordNet or Wikipedia, this limit can be overcome by means of alignment algorithms that exploit the network structure to determine the similarity of concept pairs. However, not all lexical resources provide explicit semantic relations between concepts and, hence, machine-readable dictionaries like Wiktionary have first to be transformed into semantic graphs before such graph-based approaches can be applied to them. To do this, recent work has proposed graph construction by monosemous linking, where a concept is linked to all the concepts associated with the monosemous words in its definition (Matuschek and Gurevych, 2013). However, this alignment method still involves tuning of parameters which are highly dependent on the characteristics of the generated graphs and, hence, requires hand-crafted sense alignments for the specific pair of resources to be aligned, a task which has to be replicated every time the resources are updated.

In this paper we propose a unified approach to aligning arbitrary pairs of lexical resources which is independent of their specific structure. Thanks to a novel modeling of the sense entries and an effective ontologization algorithm, our approach also fares well when resources lack relational structure or pair-specific training data is absent, meaning that it is applicable to arbitrary pairs

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 468–478,
Baltimore, Maryland, USA, June 23-25 2014.

without adaptation. We report state-of-the-art performance when aligning WordNet to Wikipedia, OmegaWiki and Wiktionary.

2 Resource Alignment

Preliminaries. Our approach for aligning lexical resources exploits the graph structure of each resource. Therefore, we assume that a lexical resource L can be represented as an undirected graph $G = (V, E)$ where V is the set of nodes, i.e., the concepts defined in the resource, and E is the set of undirected edges, i.e., semantic relations between concepts. Each concept $c \in V$ is associated with a set of lexicalizations $\mathcal{L}_G(c) = \{w_1, w_2, ..., w_n\}$. For instance, WordNet can be readily represented as an undirected graph G whose nodes are synsets and edges are modeled after the relations between synsets defined in WordNet (e.g., hypernymy, meronymy, etc.), and $\mathcal{L}_G$ is the mapping between each synset node and the set of synonyms which express the concept. However, other resources such as Wiktionary do not provide semantic relations between concepts and, therefore, have first to be transformed into semantic networks before they can be aligned using our alignment algorithm. We explain in Section 3 how a semi-structured resource which does not exhibit a graph structure can be transformed into a semantic network.

Alignment algorithm. Given a pair of lexical resources L_1 and L_2, we align each concept in L_1 by mapping it to its corresponding concept(s) in the target lexicon L_2. Algorithm 1 formalizes the alignment process: the algorithm takes as input the semantic graphs G_1 and G_2 corresponding to the two resources, as explained above, and produces as output an alignment in the form of a set A of concept pairs. The algorithm iterates over all concepts $c_1 \in V_1$ and, for each of them, obtains the set of concepts $C \subset V_2$, which can be considered as alignment candidates for c_1 (line 3). For a concept c_1, alignment candidates in G_2 usually consist of every concept $c_2 \in V_2$ that shares at least one lexicalization with c_1 in the same part of speech tag, i.e., $\mathcal{L}_{G_1}(c_1) \cap \mathcal{L}_{G_2}(c_2) \neq \emptyset$ (Reiter et al., 2008; Meyer and Gurevych, 2011). Once the set of target candidates C for a source concept c_1 is obtained, the alignment task can be cast as that of identifying those concepts in C to which c_1 should be aligned. To do this, the algorithm calculates the similarity between c_1 and each $c_2 \in C$ (line 5). If their similarity score exceeds a certain value denoted by θ

Algorithm 1 Lexical Resource Aligner

Input: graphs $H = (V_H, E_H)$, $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, the similarity threshold θ, and the combination parameter β
Output: A, the set of all aligned concept pairs

```
1: A ← ∅
2: for each concept c1 ∈ V1
3:     C ← getCandidates(c1, V2)
4:     for each concept c2 ∈ C
5:         sim ← calculateSimilarity(H, G1, G2, c1, c2, β)
6:         if sim > θ then
7:             A ← A ∪ {(c1, c2)}
8: return A
```

(line 6), the two concepts c_1 and c_2 are aligned and the pair (c_1, c_2) is added to A (line 7).

Different resource alignment techniques usually vary in the way they compute the similarity of a pair of concepts across two resources (line 5 in Algorithm 1). In the following, we present our novel approach for measuring the similarity of concept pairs.

2.1 Measuring the Similarity of Concepts

Figure 1 illustrates the procedure underlying our cross-resource concept similarity measurement technique. As can be seen, the approach consists of two main components: *definitional similarity* and *structural similarity*. Each of these components gets, as its input, a pair of concepts belonging to two different semantic networks and produces a similarity score. These two scores are then combined into an overall score (part (e) of Figure 1) which quantifies the semantic similarity of the two input concepts c_1 and c_2.

The definitional similarity component computes the similarity of two concepts in terms of the similarity of their definitions, a method that has also been used in previous work for aligning lexical resources (Niemann and Gurevych, 2011; Henrich et al., 2012). In spite of its simplicity, the mere calculation of the similarity of concept definitions provides a strong baseline, especially for cases where the definitional texts for a pair of concepts to be aligned are lexically similar, yet distinguishable from the other definitions. However, as mentioned in the introduction, definition similarity-based techniques fail at identifying the correct alignments in cases where different wordings are used or definitions are not of high quality. The structural similarity component, instead, is a novel graph-based similarity measurement technique which calculates the similarity between a pair of concepts across the semantic networks of the two resources by leveraging the semantic

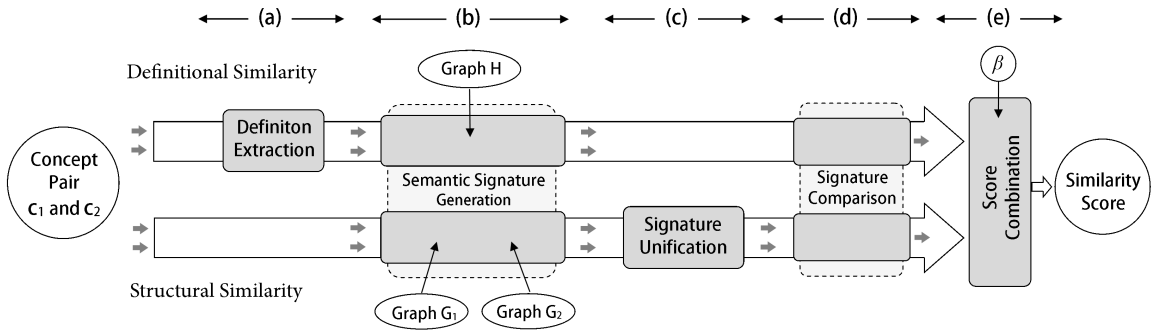


Figure 1: The process of measuring the similarity of a pair of concepts across two resources. The method consists of two components: definitional and structural similarities, each measuring a similarity score for the given concept pair. The two scores are combined by means of parameter β in the last stage.

structure of those networks. This component goes beyond the surface realization of concepts, thus providing a deeper measure of concept similarity.

The two components share the same backbone (parts (b) and (d) of Figure 1), but differ in some stages (parts (a) and (c) in Figure 1). In the following, we explain all the stages involved in the two components (gray blocks in the figure).

2.1.1 Semantic signature generation

The aim of this stage is to model a given concept or set of concepts through a vectorial semantic representation, which we refer to as the **semantic signature** of the input. We utilized Personalized PageRank (Haveliwala, 2002, PPR), a random walk graph algorithm, for calculating semantic signatures. The original PageRank (PR) algorithm (Brin and Page, 1998) computes, for a given graph, a single vector wherein each node is associated with a weight denoting its structural importance in that graph. PPR is a variation of PR where the computation is biased towards a set of initial nodes in order to capture the notion of importance with respect to those particular nodes. PPR has been previously used in a wide variety of tasks such as definition similarity-based resource alignment (Niemann and Gurevych, 2011), textual semantic similarity (Hughes and Ramage, 2007; Pilehvar et al., 2013), Word Sense Disambiguation (Agirre and Soroa, 2009; Faralli and Navigli, 2012) and semantic text categorization (Navigli et al., 2011). When applied to a semantic graph by initializing the random walks from a set of concepts (nodes), PPR yields a vector in which each concept is associated with a weight denoting its semantic relevance to the initial concepts.

Formally, we first represent a semantic network consisting of N concepts as a row-stochastic transition matrix $\mathbf{M} \in \mathbb{R}^{N \times N}$. The cell (i, j) in the matrix denotes the probability of moving from a concept i to j in the graph: 0 if no edge exists from i to j and $1/degree(i)$ otherwise. Then the PPR vector, hence the semantic signature $\mathcal{S}_{\mathbf{v}}$ of vector $\mathbf{v}$ is the unique solution to the linear system: $\mathcal{S}_{\mathbf{v}} = (1 - \alpha)\,\mathbf{v} + \alpha\,\mathbf{M}\,\mathcal{S}_{\mathbf{v}}$, where $\mathbf{v}$ is the personalization vector of size N in which all the probability mass is put on the concepts for which a semantic signature is to be computed and α is the damping factor, which is usually set to 0.85 (Brin and Page, 1998). We used the UKB[1] off-the-shelf implementation of PPR.

Definitional similarity signature. In the definitional similarity component, the two concepts c_1 and c_2 are first represented by their corresponding definitions d_1 and d_2 in the respective resources L_1 and L_2 (Figure 1(a), top). To improve expressiveness, we follow Niemann and Gurevych (2011) and further extend d_i with all the word forms associated with concept c_i and its neighbours, i.e., the union of all lexicalizations $\mathcal{L}_{G_i}(x)$ for all concepts $x \in \{c' \in V_i : (c, c') \in E_i\} \cup \{c\}$, where E_i is the set of edges in G_i. In this component the personalization vector $\mathbf{v}_i$ is set by uniformly distributing the probability mass over the nodes corresponding to the senses of all the content words in the extended definition of d_i according to the sense inventory of a semantic network H. We use the same semantic graph H for computing the semantic signatures of both definitions. Any semantic network with a dense relational structure, providing good coverage of the words appearing in the definitions, is a suitable candidate for H. For this purpose we used the WordNet (Fellbaum, 1998) graph which was further enriched by connecting

[1] http://ixa2.si.ehu.es/ukb/

each concept to all the concepts appearing in its disambiguated gloss.[2]

Structural similarity signature. In the structural similarity component (Figure 1(b), bottom), the semantic signature for each concept c_i is computed by running the PPR algorithm on its corresponding graph G_i, hence a different $\mathbf{M}_i$ is built for each of the two concepts.

2.1.2 Signature unification

As mentioned earlier, semantic signatures are vectors with dimension equal to the number of nodes in the semantic graph. Since the structural similarity signatures $\mathcal{S}_{\mathbf{v}_1}$ and $\mathcal{S}_{\mathbf{v}_2}$ are calculated on different graphs and thus have different dimensions, we need to make them comparable by unifying them. We therefore propose an approach (part (c) of Figure 1) that finds a common ground between the two signatures: to this end we consider all the concepts associated with monosemous words in the two signatures as landmarks and restrict the two signatures exclusively to those common concepts. Leveraging monosemous words as bridges between two signatures is a particularly reliable technique as typically a significant portion of all words in a lexicon are monosemous.[3]

Formally, let $\mathcal{I}_G(w)$ be an inventory mapping function that maps a term w to the set of concepts which are expressed by w in graph G. Then, given two signatures $\mathcal{S}_{\mathbf{v}_1}$ and $\mathcal{S}_{\mathbf{v}_2}$, computed on the respective graphs G_1 and G_2, we first obtain the set $\mathcal{M}$ of words that are monosemous according to both semantic networks, i.e., $\mathcal{M} = \{w : |\mathcal{I}_{G_1}(w)|=1 \wedge |\mathcal{I}_{G_2}(w)|=1\}$. We then transform each of the two signatures $\mathcal{S}_{\mathbf{v}_i}$ into a new sub-signature $\mathcal{S}'_{\mathbf{v}_i}$ whose dimension is $|\mathcal{M}|$: the k^{th} component of $\mathcal{S}'_{\mathbf{v}_i}$ corresponds to the weight in $\mathcal{S}_{\mathbf{v}_i}$ of the only concept of w_k in $\mathcal{I}_{G_i}(w_k)$. As an example, assume we are given two semantic signatures computed for two concepts in WordNet and Wiktionary. Also, consider the noun *tradeoff* which is monosemous according to both these resources. Then, each of the two unified sub-signatures will contain a component whose weight is determined by the weight of the only concept associated with *tradeoff*$_n$ in the corresponding semantic signature. As a result of the unification process, we obtain a pair of equally-sized semantic signatures with comparable components.

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

[3] For instance, we calculated that more than 80% of the words in WordNet are monosemous, with over 60% of all the synsets containing at least one of them.

2.1.3 Signature comparison

Having at hand the semantic signatures for the two input concepts, we proceed to comparing them (part (d) in Figure 1). We leverage a non-parametric measure proposed by Pilehvar et al. (2013) which first transforms each signature into a list of sorted elements and then calculates the similarity on the basis of the average ranking of elements across the two lists:

$$Sim(\mathcal{S}_{\mathbf{v}_1}, \mathcal{S}_{\mathbf{v}_2}) = \frac{\sum_{i=1}^{|T|}(r_i^1 + r_i^2)^{-1}}{\sum_{i=1}^{|T|}(2i)^{-1}} \quad (1)$$

where T is the intersection of all concepts with non-zero probability in the two signatures and r_i^j is the rank of the i^{th} entry in the j^{th} sorted list. The denominator is a normalization factor to guarantee a maximum value of one. The method penalizes the differences in the higher rankings more than it does for the lower ones. The measure was shown to outperform the conventional cosine distance when comparing different semantic signatures in multiple textual similarity tasks (Pilehvar et al., 2013).

2.1.4 Score combination

Finally (part (e) of Figure 1), we calculate the overall similarity between two concepts as a linear combination of their definitional and structural similarities: $\beta\, Sim_{def}(\mathcal{S}_{\mathbf{v}_1}, \mathcal{S}_{\mathbf{v}_2}) + (1 - \beta)\, Sim_{str}(\mathcal{S}_{\mathbf{v}_1}, \mathcal{S}_{\mathbf{v}_2})$. In Section 4.2.1, we explain how we set, in our experiments, the values of β and the similarity threshold θ (cf. alignment algorithm in Section 2).

3 Lexical Resource Ontologization

In Section 2, we presented our approach for aligning lexical resources. However, the approach assumes that the input resources can be viewed as semantic networks, which seems to limit its applicability to structured resources only. In order to address this issue and hence generalize our alignment approach to any given lexical resource, we propose a method for transforming a given machine-readable dictionary into a semantic network, a process we refer to as *ontologization*.

Our ontologization algorithm takes as input a lexicon L and outputs a semantic graph $G = (V, E)$ where, as already defined in Section 2, V is the set of concepts in L and E is the set of semantic relations between these concepts. Introducing relational links into a lexicon can be achieved in different ways. A first option is to extract binary

relations between pairs of words from raw text. Both words in these relations, however, should be disambiguated according to the given lexicon (Pantel and Pennacchiotti, 2008), making the task particularly prone to mistakes due to the high number of possible sense pairings.

Here, we take an alternative approach which requires disambiguation on the target side only, hence reducing the size of the search space significantly. We first create the empty undirected graph $G_L = (V, E)$ such that V is the set of concepts in L and $E = \emptyset$. For each source concept $c \in V$ we create a bag of content words $W = \{w_1, \ldots, w_n\}$ which includes all the content words in its definition d and, if available, additional related words obtained from lexicon relations (e.g., synonyms in Wiktionary). The problem is then cast as a disambiguation task whose goal is to identify the intended sense of each word $w_i \in W$ according to the sense inventory of L: if w_i is monosemous, i.e., $|\{\mathcal{I}_{G_L}(w_i)\}| = 1$, we connect our source concept c to the only sense c_{w_i} of w_i and set $E := E \cup \{\{c, c_{w_i}\}\}$; else, w_i has multiple senses in L. In this latter case, we choose the most appropriate concept $c_i \in \mathcal{I}_{G_L}(w_i)$ by finding the maximal similarity between the definition of c and the definitions of each sense of w_i. To do this, we apply our definitional similarity measure introduced in Section 2.1. Having found the intended sense $\hat{c}_{w_i}$ of w_i, we add the edge $\{c, \hat{c}_{w_i}\}$ to E. As a result of this procedure, we obtain a semantic graph representation G for the lexicon L.

As an example, consider the 4^{th} sense of the noun *cone* in Wiktionary (i.e., $cone_n^4$) which is defined as *"The fruit of a conifer"*. The definition contains two content words: $fruit_n$ and $conifer_n$. The latter word is monosemous in Wiktionary, hence we directly connect $cone_n^4$ to the only sense of $conifer_n$. The noun *fruit*, however, has 5 senses in Wiktionary. We therefore measure the similarity between the definition of $cone_n^4$ and all the 5 definitions of *fruit* and introduce a link from $cone_n^4$ to the sense of fruit which yields the maximal similarity value (defined as *"(botany) The seed-bearing part of a plant..."*).

4 Experiments

Lexical resources. To enable a comparison with the state of the art, we followed Matuschek and Gurevych (2013) and performed an alignment of WordNet synsets (WN) to three different collaboratively-constructed resources: Wikipedia (WP), Wiktionary (WT), and OmegaWiki (OW). We utilized the DKPro software (Zesch et al., 2008; Gurevych et al., 2012) to access the information in the foregoing three resources. For WP, WT, OW we used the dump versions 20090822, 20131002, and 20131115, respectively.

Evaluation measures. We followed previous work (Navigli and Ponzetto, 2012; Matuschek and Gurevych, 2013) and evaluated the alignment performance in terms of four measures: precision, recall, F1, and accuracy. Precision is the fraction of correct alignment judgments returned by the system and recall is the fraction of alignment judgments in the gold standard dataset that are correctly returned by the system. F1 is the harmonic mean of precision and recall. We also report results for accuracy which, in addition to true positives, takes into account true negatives, i.e., pairs which are correctly judged as unaligned.

Lexicons and semantic graphs. Here, we describe how the four semantic graphs for our four lexical resources (i.e., WN, WP, WT, OW) were constructed. As mentioned in Section 2.1.1, we build the WN graph by including all the synsets and semantic relations defined in WordNet (e.g., hypernymy and meronymy) and further populate the relation set by connecting a synset to all the other synsets that appear in its disambiguated gloss. For WP, we used the graph provided by Matuschek and Gurevych (2013), constructed by directly connecting an article (concept) to all the hyperlinks in its first paragraph, together with the category links. Our WN and WP graphs have 118K and 2.8M nodes, respectively, with the average node degree being roughly 9 in both resources.

The other two resources, i.e., WT and OW, do not provide a reliable network of semantic relations, therefore we used our ontologization approach to construct their corresponding semantic graphs. We report, in the following subsection, the experiments carried out to assess the accuracy of our ontologization method, together with the statistics of the obtained graphs for WT and OW.

4.1 Ontologization Experiments

For ontologizing WT and OW, the bag of content words W is given by the content words in sense definitions and, if available, additional related words obtained from lexicon relations (see Section 3). In WT, both of these are in word surface form and hence had to be disambiguated. For OW, however, the encoded relations, though rela-

Source	Type	WT	OW
Definition	Ambiguous	76.6%	50.7%
	Unambiguous	18.3%	32.9%
Relation	Ambiguous	2.8%	-
	Unambiguous	2.3%	16.4%
Total number of edges		2.1M	255K

Table 1: The statistics of the generated graphs for WT and OW. We report the distribution of the edges across types (i.e., ambiguous and unambiguous) and sources (i.e., definitions and relations) from which candidate words were obtained.

tively small in number, are already disambiguated and, therefore, the ontologization was just performed on the definition's content words.

The resulting graphs for WT and OW contain 430K and 48K nodes, respectively, each providing more than 95% coverage of concepts, with the average node degree being around 10 for both resources. We present in Table 1, for WT and OW, the total number of edges together with their distribution across types (i.e., ambiguous and unambiguous) and sources (i.e., definitions and relations) from which candidate words were obtained.

The edges obtained from unambiguous entries are essentially sense disambiguated on both sides whereas those obtained from ambiguous terms are a result of our similarity-based disambiguation. Hence, given that a large portion of edges came from ambiguous words (see Table 1), we carried out an experiment to evaluate the accuracy of our disambiguation method. To this end, we took as our benchmark the dataset provided by Meyer and Gurevych (2010) for evaluating relation disambiguation in WT. The dataset contains 394 manually-disambiguated relations. We compared our similarity-based disambiguation approach against the state of the art on this dataset, i.e., the WKTWSD system, which is a WT relation disambiguation algorithm based on a series of rules (Meyer and Gurevych, 2012b).

Table 2 shows the performance of our disambiguation method, together with that of WKTWSD, in terms of Precision (P), Recall (R), F1, and accuracy. The "Human" row corresponds to the inter-rater F1 and accuracy scores, i.e., the upper-bound performance on this dataset, as calculated by Meyer and Gurevych (2010). As can be seen, our method proves to be very accurate, surpassing the performance of the WKTWSD system in terms of precision, F1, and accuracy. This is particularly

Approach	P	R	F1	A
WKTWSD	0.780	**0.800**	0.790	0.840
Our method	**0.852**	0.767	**0.807**	**0.857**
Human	-	-	0.890	0.910

Table 2: The performance of relation disambiguation for our similarity-based disambiguation method, as well as for the WKTWSD system.

interesting as the WKTWSD system uses a rule-based technique specific to relation disambiguation in WT, whereas our method is resource independent and can be applied to arbitrary words in the definition of any concept. We also note that the graph constructed by Meyer and Gurevych (2010) had an average node degree of around 1.

More recently, Matuschek and Gurevych (2013) leveraged monosemous linking (cf. Section 5) in order to create denser semantic graphs for OW and WT. Our approach, however, thanks to the connections obtained through ambiguous words, can provide graphs with significantly higher coverage. As an example, for WT, Matuschek and Gurevych (2013) generated a graph where around 30% of the nodes were in isolation, whereas this number drops to around 5% in our corresponding graph.

These results show that our ontologization approach can be used to obtain dense semantic graph representations of lexical resources, while at the same time preserving a high level of accuracy. Now that all the four resources are transformed into semantic graphs, we move to our alignment experiments.

4.2 Alignment Experiments

4.2.1 Experimental setup

Datasets. As our benchmark we tested on the gold standard datasets used in Matuschek and Gurevych (2013) for three alignment tasks: WordNet-Wikipedia (WN-WP), WordNet-Wiktionary (WN-WT), and WordNet-OmegaWiki (WN-OW). However, the dataset for WN-OW was originally built for the German language and, hence, was missing many English OW concepts that could be considered as candidate target alignments. We therefore fixed the dataset for the English language and reproduced the performance of previous work on the new dataset. The three datasets contained 320, 484, and 315 WN concepts that were manually mapped to their corresponding concepts in WP, WT, and OW, respectively.

Approach	Training type	WN-WP				WN-WT				WN-OW			
		P	R	F1	A	P	R	F1	A	P	R	F1	A
SB	Cross-val.	0.780	0.780	0.780	0.950	0.670	0.650	0.660	0.910	0.749	0.691	0.716	0.886
DWSA	Tuning on subset	0.750	0.670	0.710	0.930	0.680	0.270	0.390	0.890	0.651	0.372	0.473	0.830
SB+DWSA	Cross-val. + tuning	0.750	0.870	0.810	0.950	0.680	0.710	0.690	0.920	0.794	0.688	0.735	0.898
SemAlign	Unsupervised	0.709	0.929	0.805	0.943	0.642	**0.799**	0.712	0.923	0.664	**0.761**	0.709	0.872
	Tuning on subset	**0.877**	0.792	0.833	0.960	0.672	**0.799**	**0.730**	0.930	0.750	0.717	0.733	0.893
	Cross-val.	0.852	0.835	**0.840**	**0.965**	0.680	0.769	0.722	**0.931**	0.778	0.725	**0.749**	**0.900**
	Tuning on WN-WP	-	-	-	-	**0.754**	0.627	0.684	**0.931**	**0.825**	0.584	0.684	0.889
	Tuning on WN-WT	0.738	**0.934**	0.824	0.950	-	-	-	-	0.805	0.677	0.736	**0.900**
	Tuning on WN-OW	0.744	0.925	0.824	0.950	0.684	0.766	0.723	0.930	-	-	-	-

Table 3: The performance of different systems on the task of aligning WordNet to Wikipedia (WN-WP), Wiktionary (WN-WT), and OmegaWiki (WN-OW) in terms of Precision (P), Recall (R), F1, and Accuracy (A). We present results for different configurations of our system (SemAlign), together with the state of the art in definition similarity-based alignment approaches (SB) and the best configuration of the state-of-the-art graph-based system, Dijkstra-WSA (Matuschek and Gurevych, 2013, DWSA).

Configurations. Recall from Section 2 that our resource alignment technique has two parameters: the similarity threshold θ and the combination parameter β, both defined in [0, 1]. We performed experiments with three different configurations:

- *Unsupervised*, where the two parameters are set to their middle values (i.e., 0.5), hence, no tuning is performed for either of the parameters. In this case, both the definitional and structural similarity scores are treated as equally important and two concepts are aligned if their overall similarity exceeds the middle point of the similarity scale.
- *Tuning*, where we follow Matuschek and Gurevych (2013) and tune the parameters on a subset of the dataset comprising 100 items.
- *Cross-validation*, where a 5-fold cross validation is carried out to find the optimal values for the parameters, a technique used in most of the recent alignment methods (Niemann and Gurevych, 2011; Meyer and Gurevych, 2012a; Matuschek and Gurevych, 2013).

4.2.2 Results

We show in Table 3 the alignment performance of different systems on the task of aligning WN-WP, WN-WT, and WN-OW in terms of Precision (P), Recall (R), F1, and Accuracy. The SB system corresponds to the state-of-the-art definition similarity approaches for WN-WP (Niemann and Gurevych, 2011), WN-WT (Meyer and Gurevych, 2011), and WN-OW (Gurevych et al., 2012). DWSA stands for Dijkstra-WSA, the state-of-the-art graph-based alignment approach of Matuschek and Gurevych (2013). The authors also provided results for SB+Dijkstra-WSA, a hybrid system where DWSA was tuned for high precision and, in the case when no alignment target could be found, the algorithm fell back on SB judgments. We also show the results for this system as SB+DWSA in the table.

For our approach (SemAlign) we show the results of six different runs each corresponding to a different setting. The first three (middle part of the table) correspond to the results obtained with the three configurations of SemAlign: unsupervised, with tuning on subset, and cross-validation (see Section 4.2.1). In addition to these, we performed experiments where the two parameters of SemAlign were tuned on pair-independent training data, i.e., a training dataset for a pair of resources different from the one being aligned. For this setting, we used the whole dataset of the corresponding resource pair to tune the two parameters of our system. We show the results for this setting in the bottom part of the table (last three lines).

The main feature worth remarking upon is the consistency in the results across different resource pairs: the unsupervised system gains the best recall among the three configurations (with the improvement over SB+DWSA being always statistically significant[4]) whereas tuning, both on a subset or through cross-validation, consistently leads to the best performance in terms of F1 and accuracy (with the latter being statistically significant with respect to SB+DWSA on WN-WP and WN-WT).

Moreover, the unsupervised system proves to be very robust inasmuch as it provides competitive results on all the three datasets, while it surpasses the performance of SB+DWSA on WN-WT. This

[4] All significance tests are done using z-test at $p < 0.05$.

Approach	WN-WP				WN-WT				WN-OW			
	P	**R**	**F1**	**A**	**P**	**R**	**F1**	**A**	**P**	**R**	**F1**	**A**
Dijkstra-WSA	0.750	0.670	0.710	0.930	**0.680**	0.270	0.390	0.890	0.651	0.372	0.473	0.830
SemAlign$_{str}$	**0.877**	**0.788**	**0.830**	**0.959**	0.604	**0.643**	**0.623**	**0.907**	**0.654**	**0.602**	**0.627**	**0.853**

Table 4: Performance of SemAlign when using only the structural similarity component (SemAlign$_{str}$) compared to the state-of-the-art graph-based alignment approach, Dijkstra-WSA (Matuschek and Gurevych, 2013) for our three resource pairs: WordNet to Wikipedia (WN-WP), Wiktionary (WN-WT), and OmegaWiki (WN-OW).

is particularly interesting as the latter system involves tuning of several parameters, whereas SemAlign, in its unsupervised configuration, does not need any training data nor does it involve any tuning. In addition, as can be seen in the table, SemAlign benefits from pair-independent training data in most cases across the three resource pairs with performance surpassing that of SB+DWSA, a system which is dependent on pair-specific training data. The consistency in the performance of SemAlign in its different configurations and across different resource pairs indicates its robustness and shows that our system can be utilized effectively for aligning any pair of lexical resources, irrespective of their structure or availability of training data.

The system performance is generally higher on the alignment task for WP compared to WT and OW. We attribute this difference to the dictionary nature of the latter two, where sense distinctions are more fine-grained, as opposed to the relatively concrete concepts in the WP encyclopedia.

4.3 Similarity Measure Analysis

We explained in Section 2.1 that our concept similarity measure consists of two components: the definitional and the structural similarities. Measuring the similarity of two concepts in terms of their definitions has been investigated in previous work (Niemann and Gurevych, 2011; Henrich et al., 2012). The structural similarity component of our approach, however, is novel, but at the same time one of the very few measures which enables the computation of the similarity of concepts across two resources directly and independently of the similarity of their definitions. A comparable approach is the Dijkstra-WSA proposed by Matuschek and Gurevych (2013) which, as also mentioned earlier in the Introduction, first connects the two resources' graphs by leveraging monosemous linking and then aligns two concepts across the two graphs on the basis of their shortest distance. To gain more insight into the effectiveness of our structural similarity measure in comparison to the Dijkstra-WSA method, we carried out an experiment where our alignment system used only the structural similarity component, a variant of our system we refer to as SemAlign$_{str}$. Both systems (i.e., SemAlign$_{str}$ and Dijkstra-WSA) were tuned on 100-item subsets of the corresponding datasets.

We show in Table 4 the performance of the two systems on our three datasets. As can be seen in the table, SemAlign$_{str}$ consistently improves over Dijkstra-WSA according to recall, F1 and accuracy with all the differences in recall and accuracy being statistically significant ($p < 0.05$). The improvement is especially noticeable for pairs involving either WT or OW where, thanks to the relatively denser semantic graphs obtained by means of our ontologization technique, the gap in F1 is about 0.23 (WN-WT) and 0.15 (WN-OW).

In addition, as we mentioned earlier, for WN-WP we used the same graph as that of Dijkstra-WSA, since both WN and WP provide a full-fledged semantic network and thus neither needed to be ontologized. Therefore, the considerable performance improvement over Dijkstra-WSA on this resource pair shows the effectiveness of our novel concept similarity measure independently of the underlying semantic network.

5 Related Work

Resource ontologization. Having lexical resources represented as semantic networks is highly beneficial. A good example is WordNet, which has been exploited as a semantic network in dozens of NLP tasks (Fellbaum, 1998). A recent prominent case is Wikipedia (Medelyan et al., 2009; Hovy et al., 2013) which, thanks to its inter-article hyperlink structure, provides a rich backbone for structuring additional information (Auer et al., 2007; Suchanek et al., 2008; Moro and Navigli, 2013; Flati et al., 2014). However, there are many large-scale resources, such as Wiktionary for instance, which by their very nature are not in the form of a graph. This is

usually the case with machine-readable dictionaries, where structuring the resource involves the arduous task of connecting lexicographic senses by means of semantic relations. Surprisingly, despite their vast potential, little research has been conducted on the automatic ontologization of collaboratively-constructed dictionaries like Wiktionary and OmegaWiki. Meyer and Gurevych (2012a) and Matuschek and Gurevych (2013) provided approaches for building graph representations of Wiktionary and OmegaWiki. The resulting graphs, however, were either sparse or had a considerable portion of the nodes left in isolation. Our approach, in contrast, aims at transforming a lexical resource into a full-fledged semantic network, hence providing a denser graph with most of its nodes connected.

Resource alignment. Aligning lexical resources has been a very active field of research in the last decade. One of the main objectives in this area has been to enrich existing ontologies by means of complementary information from other resources. As a matter of fact, most efforts have been concentrated on aligning the *de facto* community standard sense inventory, i.e. WordNet, to other resources. These include: the Roget's thesaurus and Longman Dictionary of Contemporary English (Kwong, 1998), FrameNet (Laparra and Rigau, 2009), VerbNet (Shi and Mihalcea, 2005) or domain-specific terminologies such as the Unified Medical Language System (Burgun and Bodenreider, 2001). More recently, the growth of collaboratively-constructed resources has seen the development of alignment approaches with Wikipedia (Ruiz-Casado et al., 2005; Auer et al., 2007; Suchanek et al., 2008; Reiter et al., 2008; Navigli and Ponzetto, 2012), Wiktionary (Meyer and Gurevych, 2011) and OmegaWiki (Gurevych et al., 2012). Last year Matuschek and Gurevych (2013) proposed Dijkstra-WSA, a graph-based approach relying on shortest paths between two concepts when the two corresponding resources graphs were combined by leveraging monosemous linking. Their method when backed off with other definition similarity based approaches (Niemann and Gurevych, 2011; Meyer and Gurevych, 2011), achieved state-of-the-art results on the mapping of WordNet to different collaboratively-constructed resources. This approach, however, in addition to setting the threshold for the definition similarity component by means of cross validation, also required other parameters to be tuned, such as the allowed path length (λ) and the maximum number of edges in a graph. The optimal value for the λ parameter varied from one resource pair to another, and even for a specific resource pair it had to be tuned for each configuration. This made the approach dependent on the training data for the specific pair of resources that were to be aligned. Instead of measuring the similarity of two concepts on the basis of their distance in the combined graph, our approach models each concept through a rich vectorial representation we refer to as semantic signature and compares the two concepts in terms of the similarity of their semantic signatures. This rich representation leads to our approach having a good degree of robustness such that it can achieve competitive results even in the absence of training data. This enables our system to be applied effectively for aligning new pairs of resources for which no training data is available, with state-of-the-art performance.

6 Conclusions

This paper presents a unified approach for aligning lexical resources. Our method leverages a novel similarity measure which enables a direct structural comparison of concepts across different lexical resources. Thanks to an effective ontologization method, our alignment approach can be applied to any pair of lexical resources independently of whether they provide a full-fledged network structure. We demonstrate that our approach achieves state-of-the-art performance on aligning WordNet to three collaboratively-constructed resources with different characteristics, i.e., Wikipedia, Wiktionary, and OmegaWiki. We also show that our approach is robust across its different configurations, even when the training data is absent, enabling it to be used effectively for aligning new pairs of lexical resources for which no resource-specific training data is available. In future work, we plan to extend our concept similarity measure across different natural languages. We release all our data at `http://lcl.uniroma1.it/semalign`.

Acknowledgments

The authors gratefully acknowledge the support of the ERC Starting Grant MultiJEDI No. 259234.

We would like to thank Michael Matuschek for providing us with Wikipedia graphs and alignment datasets.

References

Eneko Agirre and Aitor Soroa. 2009. Personalizing PageRank for Word Sense Disambiguation. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 33–41, Athens, Greece.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ive. 2007. DBpedia: A nucleus for a web of open data. In *Proceedings of 6th International Semantic Web Conference joint with 2nd Asian Semantic Web Conference (ISWC+ASWC 2007)*, pages 722–735, Busan, Korea.

Sergey Brin and Michael Page. 1998. Anatomy of a large-scale hypertextual Web search engine. In *Proceedings of the 7^{th} Conference on World Wide Web*, pages 107–117, Brisbane, Australia.

Anita Burgun and Olivier Bodenreider. 2001. Comparing terms, concepts and semantic classes in WordNet and the Unified Medical Language System. In *Proceedings of NAACL Workshop, WordNet and Other Lexical Resources: Applications, Extensions and Customizations*, pages 77–82, Pittsburgh, USA.

Gerard de Melo and Gerhard Weikum. 2010. Providing multilingual, multimodal answers to lexical database queries. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, pages 348–355, Valletta, Malta.

Stefano Faralli and Roberto Navigli. 2012. A New Minimally-supervised Framework for Domain Word Sense Disambiguation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1411–1422, Jeju, Korea.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Database*. MIT Press, Cambridge, MA.

Tiziano Flati, Daniele Vannella, Tommaso Pasini, and Roberto Navigli. 2014. Two is bigger (and better) than one: the Wikipedia Bitaxonomy Project. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)*, Baltimore, Maryland.

Iryna Gurevych, Judith Eckle-Kohler, Silvana Hartmann, Michael Matuschek, Christian M. Meyer, and Christian Wirth. 2012. UBY - a large-scale unified lexical-semantic resource based on LMF. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 580–590, Avignon, France.

Taher H. Haveliwala. 2002. Topic-sensitive PageRank. In *Proceedings of the 11th international conference on World Wide Web*, pages 517–526, Hawaii, USA.

Verena Henrich, Erhard Hinrichs, and Tatiana Vodolazova. 2011. Semi-automatic extension of GermaNet with sense definitions from Wiktionary. In *Proceedings of 5th Language & Technology Conference (LTC 2011)*, pages 126–130, Pozna, Poland.

Verena Henrich, Erhard W. Hinrichs, and Klaus Suttner. 2012. Automatically linking GermaNet to Wikipedia for harvesting corpus examples for GermaNet senses. *In Journal for Language Technology and Computational Linguistics (JLCL)*, 27(1):1–19.

Eduard H. Hovy, Roberto Navigli, and Simone Paolo Ponzetto. 2013. Collaboratively built semi-structured content and Artificial Intelligence: The story so far. *Artificial Intelligence*, 194:2–27.

Thad Hughes and Daniel Ramage. 2007. Lexical semantic relatedness with random graph walks. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing*, pages 581–589, Prague, Czech Republic.

Oi Yee Kwong. 1998. Aligning WordNet with additional lexical resources. In *COLING-ACL98 Workshop on Usage of WordNet in Natural Language Processing Systems*, pages 73–79, Montreal, Canada.

Egoitzand Laparra and German Rigau. 2009. Integrating WordNet and FrameNet using a knowledge-based Word Sense Disambiguation algorithm. In *Proceedings of Recent Advances in Natural Language Processing (RANLP09)*, pages 1–6, Borovets, Bulgaria.

Michael Matuschek and Iryna Gurevych. 2013. Dijkstra-WSA: A graph-based approach to word sense alignment. *Transactions of the Association for Computational Linguistics (TACL)*, 1:151–164.

Olena Medelyan, David Milne, Catherine Legg, and Ian H. Witten. 2009. Mining meaning from Wikipedia. *International Journal of Human-Computer Studies*, 67(9):716–754.

Christian M. Meyer and Iryna Gurevych. 2010. "worth its weight in gold or yet another resource"; a comparative study of Wiktionary, OpenThesaurus and GermaNet. In *Proceedings of the 11th International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing'10, pages 38–49, Iasi, Romania.

Christian M. Meyer and Iryna Gurevych. 2011. What psycholinguists know about Chemistry: Aligning Wiktionary and WordNet for increased domain coverage. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*, pages 883–892, Chiang Mai, Thailand.

Christian M. Meyer and Iryna Gurevych. 2012a. OntoWiktionary: Constructing an ontology from the collaborative online dictionary Wiktionary. In *Semi-Automatic Ontology Development: Processes and Resources*, pages 131–161. IGI Global.

Christian M. Meyer and Iryna Gurevych. 2012b. To exhibit is not to loiter: A multilingual, sense-disambiguated Wiktionary for measuring verb similarity. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*, pages 1763–1780, Mumbai, India.

Andrea Moro and Roberto Navigli. 2013. Integrating syntactic and semantic analysis into the Open Information Extraction paradigm. In *Proceedings of the* 23^{rd} *International Joint Conference on Artificial Intelligence (IJCAI 2013)*, pages 2148–2154, Beijing, China.

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

Roberto Navigli, Stefano Faralli, Aitor Soroa, Oier de Lacalle, and Eneko Agirre. 2011. Two birds with one stone: Learning semantic models for text categorization and Word Sense Disambiguation. In *Proceedings of the 20th ACM Conference on Information and Knowledge Management (CIKM)*, pages 2317–2320, Glasgow, UK.

Roberto Navigli. 2006. Meaningful clustering of senses helps boost word sense disambiguation performance. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics joint with the 21st International Conference on Computational Linguistics (COLING-ACL 2006)*, pages 105–112, Sydney, Australia.

Elisabeth Niemann and Iryna Gurevych. 2011. The people's web meets linguistic knowledge: Automatic sense alignment of Wikipedia and WordNet. In *Proceedings of the Ninth International Conference on Computational Semantics*, pages 205–214, Oxford, United Kingdom.

Martha Palmer, Daniel Gildea, and Nianwen Xue. 2010. *Semantic Role Labeling*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Patrick Pantel and Marco Pennacchiotti. 2008. Automatically harvesting and ontologizing semantic relations. In *Proceedings of the 2008 Conference on Ontology Learning and Population: Bridging the Gap Between Text and Knowledge*, pages 171–195, Amsterdam, The Netherlands.

Mohammad Taher Pilehvar, David Jurgens, and Roberto Navigli. 2013. Align, Disambiguate and Walk: a Unified Approach for Measuring Semantic Similarity. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1341–1351, Sofia, Bulgaria.

Nils Reiter, Matthias Hartung, and Anette Frank. 2008. A resource-poor approach for linking ontology classes to Wikipedia articles. In Johan Bos and Rodolfo Delmonte, editors, *Semantics in Text Processing*, volume 1 of *Research in Computational Semantics*, pages 381–387. College Publications, London, England.

Maria Ruiz-Casado, Enrique Alfonseca, and Pablo Castells. 2005. Automatic assignment of Wikipedia encyclopedic entries to WordNet synsets. In *Proceedings of the Third International Conference on Advances in Web Intelligence*, pages 380–386, Lodz, Poland.

Lei Shi and Rada Mihalcea. 2005. Putting pieces together: Combining FrameNet, VerbNet and WordNet for robust semantic parsing. In *Proceedings of the 6th International Conference on Computational Linguistics and Intelligent Text Processing*, pages 100–111, Mexico City, Mexico.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2008. Yago: A large ontology from Wikipedia and WordNet. *Journal of Web Semantics*, 6(3):203–217.

Torsten Zesch, Christof Müller, and Iryna Gurevych. 2008. Using Wiktionary for computing semantic relatedness. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 2*, pages 861–866, Chicago, Illinois.

Predicting the relevance of distributional semantic similarity with contextual information

Philippe Muller
IRIT, Toulouse University
Université Paul Sabatier
118 Route de Narbonne
31062 Toulouse Cedex 04
philippe.muller@irit.fr

Cécile Fabre
CLLE, Toulouse University
Université Toulouse-Le Mirail
5 alles A. Machado
31058 Toulouse Cedex
cecile.fabre@univ-tlse2.fr

Clémentine Adam
CLLE, Toulouse University
Université Toulouse-Le Mirail
5 alles A. Machado
31058 Toulouse Cedex
clementine.adam@univ-tlse2.fr

Abstract

Using distributional analysis methods to compute semantic proximity links between words has become commonplace in NLP. The resulting relations are often noisy or difficult to interpret in general. This paper focuses on the issues of evaluating a distributional resource and filtering the relations it contains, but instead of considering it in abstracto, we focus on pairs of words in context. In a discourse, we are interested in knowing if the semantic link between two items is a by-product of textual coherence or is irrelevant. We first set up a human annotation of semantic links with or without contextual information to show the importance of the textual context in evaluating the relevance of semantic similarity, and to assess the prevalence of actual semantic relations between word tokens. We then built an experiment to automatically predict this relevance, evaluated on the reliable reference data set which was the outcome of the first annotation. We show that in-document information greatly improve the prediction made by the similarity level alone.

1 Introduction

The goal of the work presented in this paper is to improve distributional thesauri, and to help evaluate the content of such resources. A distributional thesaurus is a lexical network that lists semantic neighbours, computed from a corpus and a similarity measure between lexical items, which generally captures the similarity of contexts in which the items occur. This way of building a semantic network has been very popular since (Grefenstette, 1994; Lin, 1998), even though the nature of the information it contains is hard to define, and its evaluation is far from obvious. A distributional thesaurus includes a lot of "noise" from a semantic point of view, but also lists relevant lexical pairs that escape classical lexical relations such as synonymy or hypernymy.

There is a classical dichotomy when evaluating NLP components between extrinsic and intrinsic evaluations (Jones, 1994), and this applies to distributional thesauri (Curran, 2004; Poibeau and Messiant, 2008). Extrinsic evaluations measure the capacity of a system in which a resource or a component to evaluate has been used, for instance in this case information retrieval (van der Plas, 2008) or word sense disambiguation (Weeds and Weir, 2005). Intrinsic evaluations try to measure the resource itself with respect to some human standard or judgment, for instance by comparing a distributional resource with respect to an existing synonym dictionary or similarity judgment produced by human subjects (Pado and Lapata, 2007; Baroni and Lenci, 2010). The shortcomings of these methods have been underlined in (Baroni and Lenci, 2011). Lexical resources designed for other objectives put the spotlight on specific areas of the distributional thesaurus. They are not suitable for the evaluation of the whole range of semantic relatedness that is exhibited by distributional similarities, which exceeds the limits of classical lexical relations, even though researchers have tried to collect equivalent resources manually, to be used as a gold standard (Weeds, 2003; Bordag, 2008; Anguiano et al., 2011). One advantage of distributional similarities is to exhibit a lot of different semantic relations, not necessarily standard lexical relations. Even with respect to established lexical resources, distributional approaches may improve coverage, complicating the evaluation even more.

The method we propose here has been designed as an intrinsic evaluation with a view to validate semantic proximity links in a broad per-

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 479–488,
Baltimore, Maryland, USA, June 23-25 2014.

spective, to cover what (Morris and Hirst, 2004) call "non classical lexical semantic relations". For instance, agentive relations (author/publish, author/publication) or associative relations (actor/cinema) should be considered. At the same time, we want to filter associations that can be considered as accidental in a semantic perspective (e.g. flag and composer are similar because they appear a lot with nationality names). We do this by judging the relevance of a lexical relation in a context where both elements of a lexical pair occur. We show not only that this improves the reliability of human judgments, but also that it gives a framework where this relevance can be predicted automatically. We hypothetize that evaluating and filtering semantic relations in texts where lexical items occur would help tasks that naturally make use of semantic similarity relations, but assessing this goes beyond the present work.

In the rest of this paper, we describe the resource we used as a case study, and the data we collected to evaluate its content (section 2). We present the experiments we set up to automatically filter semantic relations in context, with various groups of features that take into account information from the corpus used to build the thesaurus and contextual information related to occurrences of semantic neighbours 3). Finally we discuss some related work on the evaluation and improvement of distributional resources (section 4).

2 Evaluation of lexical similarity in context

2.1 Data

We use a distributional resource for French, built on a 200M word corpus extracted from the French Wikipedia, following principles laid out in (Bourigault, 2002) from a structured model (Baroni and Lenci, 2010), i.e. using syntactic contexts. In this approach, contexts are triples (governor,relation,dependent) derived from syntactic dependency structures. Governors and dependents are verbs, adjectives and nouns. Multiword units are available, but they form a very small subset of the resulting neighbours. Base elements in the thesaurus are of two types: arguments (dependents' lemma) and predicates (governor+relation). This is to keep the predicate/argument distinction since similarities will be computed between predicate pairs or argument pairs, and a lexical item can appear in many predicates and as an argument (e.g. *interest* as argument, *interest_for* as one predicate). The similarity of distributions was computed with Lin's score (Lin, 1998).

We will talk of lexical neighbours or distributional neighbours to label pairs of predicates or arguments, and in the rest of the paper we consider only lexical pairs with a Lin score of at least 0.1, which means about 1.4M pairs. This somewhat arbitrary level is an *a priori* threshold to limit the resulting database, and it is conservative enough not to exclude potential interesting relations. The distribution of scores is given figure 1; 97% of the selected pairs have a score between 0.1 and 0.29.

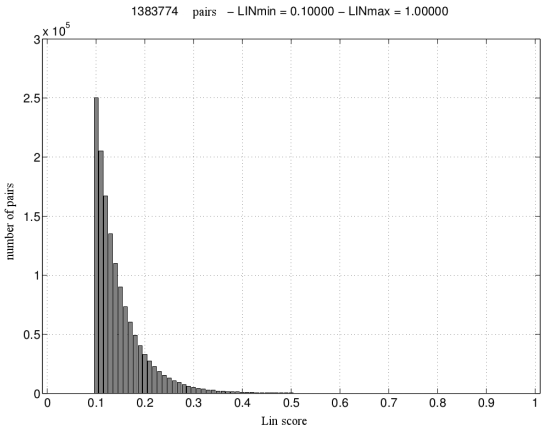


Figure 1: Histogram of Lin scores for pairs considered.

To ease the use of lexical neighbours in our experiments, we merged together predicates that include the same lexical unit, *a posteriori*. Thus there is no need for a syntactic analysis of the context considered when exploiting the resource, and sparsity is less of an issue[1].

2.2 Annotation

In order to evaluate the resource, we set up an annotation in context: pairs of lexical items are to be judged in their context of use, in texts where they occur together. To verify that this methodology is useful, we did a preliminary annotation to contrast judgment on lexical pairs with or without this contextual information. Then we made a larger annotation in context once we were assured of the reliability of the methodology.

For the preliminary test, we asked three annotators to judge the similarity of pairs of lexical items without any context (no-context), and to judge the

[1]Whenever two predicates with the same lemma have common neighbours, we average the score of the pairs.

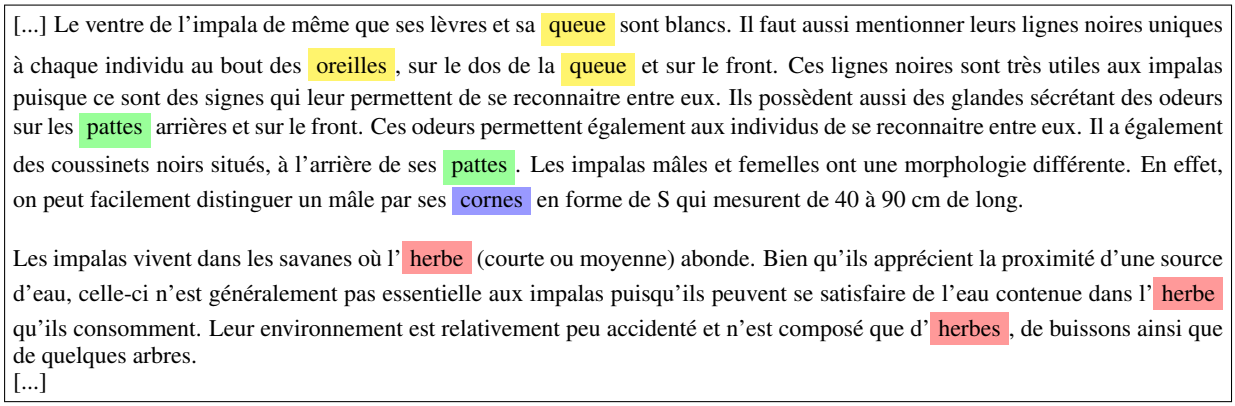


Figure 2: Example excerpt during the annotation of lexical pairs: annotators focus on a target item (here *corne*, horn, in blue) and must judge yellow words (pending: *oreille/queue*, ear/tail), either validating their relevance (green words: *pattes*, legs) or rejecting them (red words: *herbe*, grass). The text describes the morphology of the impala, and its habitat.

similarity of pairs presented within a paragraph where they both occur (in context). The three annotators were linguists, and two of them (1 and 3) knew about the resource and how it was built. For each annotation, 100 pairs were randomly selected, with the following constraints:

- for the no-context annotation, candidate pairs had a Lin score above 0.2, which placed them in the top 14% of lexical neighbours with respect to the similarity level.
- for the in context annotation, the only constraint was that the pairs occur in the same paragraph somewhere in the corpus used to build the resource. The example paragraph was chosen at random.

The guidelines given in both cases were the same: "Do you think the two words are semantically close ? In other words, is there a semantic relation between them, either classical (synonymy, hypernymy, co-hyponymy, meronymy, co-meronymy) or not (the relation can be paraphrased but does not belong to the previous cases) ?"

For the pre-test, agreement was rather moderate without context (the average of pairwise kappas was .46), and much better with a context (average = .68), with agreement rates above 90%. This seems to validate the feasability of a reliable annotation of relatedness in context, so we went on for a larger annotation with two of the previous annotators.

For the larger annotation, the protocol was slightly changed: two annotators were given 42 full texts from the original corpus where lexical neighbours occurred. They were asked to judge the relation between two items types, regardless of the number of occurrences in the text. This time there was no filtering of the lexical pairs beyond the 0.1 threshold of the original resource. We followed the well-known postulate (Gale et al., 1992) that all occurrences of a word in the same discourse tend to have the same sense ("one sense per discourse"), in order to decrease the annotator workload. We also assumed that the relation between these items remain stable within the document, an arguably strong hypothesis that needed to be checked against inter-annotator agreement before beginning the final annotation . It turns out that the kappa score (0.80) shows a better inter-annotator agreement than during the preliminary test, which can be explained by the larger context given to the annotator (the whole text), and thus more occurrences of each element in the pair to judge, and also because the annotators were more experienced after the preliminary test. Agreement measures are summed-up table 1. An excerpt of an example text, as it was presented to the annotators, is shown figure 2.

Overall, it took only a few days to annotate 9885 pairs of lexical items. Among the pairs that were presented to the annotators, about 11% were judged as relevant by the annotators. It is not easy to decide if the non-relevant pairs are just noise, or context-dependent associations that were not present in the actual text considered (for polysemy reasons for instance), or just low-level associations. An important aspect is thus to guarantee that there is a correlation between the sim-

Annotators	Non-contextual		Contextual	
	Agreement rate	Kappa	Agreement rate	Kappa
N1+N2	77%	0.52	91%	0.66
N1+N3	70%	0.36	92%	0.69
N2+N3	79%	0.50	92%	0.69
Average	75,3%	0,46	91,7%	0,68
Experts	NA	NA	90.8%	0.80

Table 1: Inter-annotator agreements with Cohen's Kappa for contextual and non-contextual annotations. N1, N2, N3 were annotators during the pre-test; expert annotation was made on a different dataset from the same corpus, only with the full discourse context.

ilarity score (Lin's score here), and the evaluated relevance of the neighbour pairs. Pearson correlation factor shows that Lin score is indeed significantly correlated to the annotated relevance of lexical pairs, albeit not strongly ($r = 0.159$).

The produced annotation[2] can be used as a reference to explore various aspects of distributional resources, with the caveat that it is as such a bit dependent on the particular resource used. We nonetheless assume that some of the relevant pairs would appear in other thesauri, or would be of interest in an evaluation of another resource.

The first thing we can analyse from the annotated data is the impact of a threshold on Lin's score to select relevant lexical pairs. The resource itself is built by choosing a cut-off which is supposed to keep pairs with a satisfactory similarity, but this threshold is rather arbitrary. Figure 3 shows the influence of the threshold value to select relevant pairs, when considering precision and recall of the pairs that are kept when choosing the threshold, evaluated against the human annotation of relevance in context. In case one wants to optimize the F-score (the harmonic mean of precision and recall) when extracting relevant pairs, we can see that the optimal point is at .24 for a threshold of .22 on Lin's score. This can be considered as a baseline for extraction of relevant lexical pairs, to which we turn in the following section.

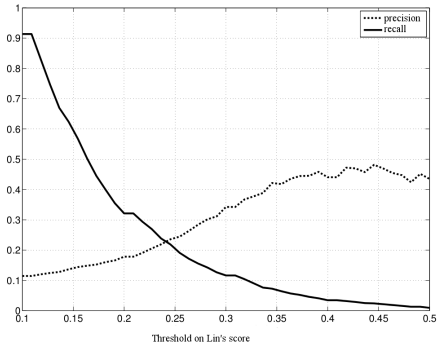


Figure 3: Precision and recall on relevant links with respect to a threshold on the similarity measure (Lin's score)

3 Experiments: predicting relevance in context

The outcome of the contextual annotation presented above is a rather sizeable dataset of validated semantic links, and we showed these linguistic judgments to be reliable. We used this dataset to set up a supervised classification experiment in order to automatically predict the relevance of a semantic link in a given discourse. We present now the list of features that were used for the model. They can be divided in three groups, according to their origin: they are computed from the whole corpus, gathered from the distributional resource, or extracted from the considered text which contains the semantic pair to be evaluated.

3.1 Features

For each pair *neighbour*$_a$/*neighbour*$_b$, we computed a set of features from Wikipedia (the corpus used to derive the distributional similarity): We first computed the frequencies of each item in the corpus, $freq_a$ and $freq_b$, from which we derive

- $freq_{min}$, $freq_{max}$: the min and max of $freq_a$ and $freq_b$;
- $freq_{\times}$: the combination of the two, or $\log(freq_a \times freq_b)$

[2]Freely available here `http://www.irit.fr/~Philippe.Muller/resources.html`.

We also measured the syntagmatic association of *neighbour*$_a$ and *neighbour*$_b$, with a mutual information measure (Church and Hanks, 1990), computed from the cooccurrence of two tokens within the same paragraph in Wikipedia. This is a rather large window, and thus gives a good coverage with respect to the neighbour database (70% of all pairs).

A straightforward parameter to include to predict the relevance of a link is of course the similarity measure itself, here Lin's information measure. But this can be complemented by additional information on the similarity of the neighbours, namely:

- each neighbour productivity : $prod_a$ and $prod_b$ are defined as the numbers of neighbours of respectively *neighbour*$_a$ and *neighbour*$_b$ in the database (thus related tokens with a similarity above the threshold), from which we derive three features as for frequencies: the min, the max, and the log of the product. The idea is that neighbours whith very high productivity give rise to less reliable relations.

- the ranks of tokens in other related items neighbours: $rank_{a-b}$ is defined as the rank of *neighbour*$_a$ among neighbours of *neighbour*$_b$ ordered with respect to Lin's score; $rang_{b-a}$ is defined similarly and again we consider as features the min, max and log-product of these ranks.

We add two categorial features, of a more linguistic nature:

- $cats$ is the pair of part-of-speech for the related items, e.g. to distinguish the relevance of NN or VV pairs.

- $predarg$ is related to the predicate/argument distinction: are the related items predicates or arguments ?

The last set of features derive from the occurrences of related tokens in the considered discourses:

First, we take into account the frequencies of items within the text, with three features as before: the min of the frequencies of the two related items, the max, and the log-product. Then we consider a *tf·idf* (Salton et al., 1975) measure, to evaluate the specificity and arguably the importance of a word

Feature	Description
$freq_{\min}$	$\min(freq_a, freq_b)$
$freq_{\max}$	$\max(freq_a, freq_b)$
$freq_{\times}$	$\log(freq_a \times freq_b)$
im	$im = \log \frac{P(a,b)}{P(a) \cdot P(b)}$
lin	Lin's score
$rank_{\min}$	$\min(rank_{a-b}, rank_{b-a})$
$rank_{\max}$	$\max(rank_{a-b}, rank_{b-a})$
$rank_{\times}$	$\log(rank_{a-b} \times rank_{b-a})$
$prod_{\min}$	$\min(prod_a, prod_b)$
$prod_{\max}$	$\max(prod_a, prod_b)$
$prod_{\times}$	$\log(prod_a \times prod_b)$
$cats$	neighbour pos pair
$predarg$	predicate or argument
$freqtxt_{\min}$	$\min(freqtxt_a, freqtxt_b)$
$freqtxt_{\max}$	$\max(freqtxt_a, freqtxt_b)$
$freqtxt_{\times}$	$\log(freqtxt_a \times freqstxt_b)$
tf·ipf	*tf·ipf*(*neighbour*$_a$)×*tf·ipf*(*neighbour*$_b$)
$copr_{ph}$	copresence in a sentence
$copr_{para}$	copresence in a paragraph
sd	smallest distance between *neighbour*$_a$ and *neighbour*$_b$
gd	highest distance between *neighbour*$_a$ and *neighbour*$_b$
ad	average distance between *neighbour*$_a$ and *neighbour*$_b$
$prodtxt_{\min}$	$\min(prod_a, prod_b)$
$prodtxt_{\max}$	$\max(prod_a, prod_b)$
$prodtxt_{\times}$	$\log(prod_a \times prod_b)$
cc	belong to the same lexical connected component

Table 2: Summary of features used in the supervised model, with respect to two lexical items a and b. The first group is corpus related, the second group is related to the distributional database, the third group is related to the textual context. Freq is related to the frequencies in the corpus, Freqtext the frequencies in the considered text.

in a document or within a document. Several variants of *tf·idf* have been proposed to adapt the measure to more local areas in a text with respect to the whole document. For instance (Dias et al., 2007) propose a *tf·isf*(*term frequency · inverse sentence frequency*), for topic segmentation. We similarly defined a *tf·ipf* measure based on the frequency of a word within a paragraph with respect to its frequency within the text. The resulting feature we used is the product of this measure for $neighbour_a$ and $neighbour_b$.

A few other contextual features are included in the model: the distances between pairs of related items, instantiated as:

- distance in words between occurrences of related word types:
  - minimal distance between two occurrences (sd)
  - maximal distance between two occurrences (gd)
  - average distance (ad) ;
- boolean features indicating whether $neighbour_a$ and $neighbour_b$ appear in the same sentence ($copr_s$) or the same paragraph ($copr_{para}$).

Finally, we took into account the network of related lexical items, by considering the largest sets of words present in the text and connected in the database (self-connected components), by adding the following features:

- the degree of each lemma, seen as a node in this similarity graph, combined as above in minimal degree of the pair, maximal degree, and product of degrees ($prodtxt_{\min}$, $prodtxt_{\max}$, $prodtxt_{\times}$). This is the number of pairs (present in the text) where a lemma appears in.
- a boolean feature cc saying whether a lexical pair belongs to a connected component of the text, except the largest. This reflects the fact that a small component may concern a lexical field which is more specific and thus more relevant to the text.

  Figure 4 shows examples of self-connected components in an excerpt of the page on *Gorille* (gorilla), e.g. the set {*pelage, dos, fourrure*} (coat, back, fur).

The last feature is probably not entirely independent from the productivity of an item, or from the tf.ipf measure.

Table 2 sums up the features used in our model.

3.2 Model

Our task is to identify relevant similarities between lexical items, between all possible related pairs, and we want to train an inductive model, a classifier, to extract the relevant links. We have seen that the relevant/not relevant classification is very imbalanced, biased towards the "not relevant" category (about 11%/89%), so we applied methods dedicated to counter-balance this, and will focus on the precision and recall of the predicted relevant links.

Following a classical methodology, we made a 10-fold cross-validation to evaluate robustly the performance of the classifiers. We tested a few popular machine learning methods, and report on two of them, a naive bayes model and the best method on our dataset, the Random Forest classifier (Breiman, 2001). Other popular methods (maximum entropy, SVM) have shown slightly inferior combined F-score, even though precision and recall might yield more important variations. As a baseline, we can also consider a simple threshold on the lexical similarity score, in our case Lin's measure, which we have shown to yield the best F-score of 24% when set at 0.22.

To address class imbalance, two broad types of methods can be applied to help the model focus on the minority class. The first one is to resample the training data to balance the two classes, the second one is to penalize differently the two classes during training when the model makes a mistake (a mistake on the minority class being made more costly than on the majority class). We tested the two strategies, by applying the classical Smote method of (Chawla et al., 2002) as a kind of resampling, and the ensemble method MetaCost of (Domingos, 1999) as a cost-aware learning method. Smote synthetizes and adds new instances similar to the minority class instances and is more efficient than a mere resampling. MetaCost is an interesting meta-learner that can use any classifier as a base classifier. We used Weka's implementations of these methods (Frank et al., 2004), and our experiments and comparisons are thus easily replicated on our dataset, provided with this paper, even though they can be improved by

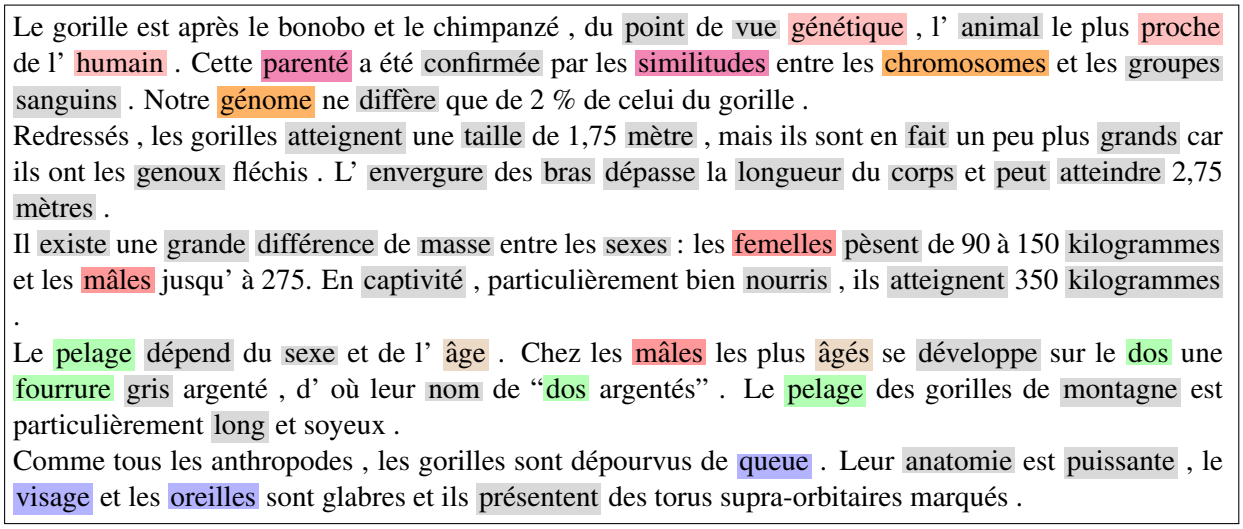


Figure 4: A few connected lexical components of the similarity graph, projected on a text, each in a different color. The groups are, in order of appearance of the first element: {genetic, close, human}, {similarity, kinship}, {chromosome, genome}, {male, female}, {coat, back, fur}, {age/N, aged/A}, {ear, tail, face}. The text describes the gorilla species, more particularly its morphology. Gray words are other lexical elements in the neighbour database.

refinements of these techniques. We chose the following settings for the different models: naive bayes uses a kernel density estimation for numerical features, as this generally improves performance. For Random Forests, we chose to have ten trees, and each decision is taken on a randomly chosen set of five features. For resampling, Smote advises to double the number of instances of the minority class, and we observed that a bigger resampling degrades performances. For cost-aware learning, a sensible choice is to invert the class ratio for the cost ratio, i.e. here the cost of a mistake on a relevant link (false negative) is exactly 8.5 times higher than the cost on a non-relevant link (false positive), as non-relevant instances are 8.5 times more present than relevant ones.

3.3 Results

We are interested in the precision and recall for the "relevant" class. If we take the best simple classifier (random forests), the precision and recall are 68.1% and 24.2% for an F-score of 35.7%, and this is significantly beaten by the Naive Bayes method as precision and recall are more even (F-score of 41.5%). This is already a big improvement on the use of the similarity measure alone (24%). Also note that predicting every link as relevant would result in a 2.6% precision, and thus a 5% F-score. The random forest model is significantly improved by the balancing techniques: the overall best F-score of 46.3% is reached with Random Forests and the cost-aware learning method. Table 3 sums up the scores for the different configurations, with precision, recall, F-score and the confidence interval on the F-score. We analysed the learning curve by doing a cross-validation on reduced set of instances (from 10% to 90%); F1-scores range from 37.3% with 10% of instances and stabilize at 80%, with small increment in every case.

The filtering approach we propose seems to yield good results, by augmenting the similarity built on the whole corpus with signals from the local contexts and documents where related lexical items appear together.

To try to analyse the role of each set of features, we repeated the experiment but changed the set of features used during training, and results are shown table 4 for the best method (RF with cost-aware learning).

We can see that similarity-related features (measures, ranks) have the biggest impact, but the other ones also seem to play a significant role. We can draw the tentative conclusion that the quality of distributional relations depends on the contextualizing of the related lexical items, beyond just the similarity score and the ranks of items as neighbours of other items.

Method	Precision	Recall	F-score	CI
Baseline (Lin threshold)	24.0	24.0	24.0	
RF	**68.1**	24.2	35.7	± 3.4
NB	34.8	51.3	41.5	± 2.6
RF+resampling	56.6	32.0	40.9	± 3.3
NB+resampling	32.8	54.0	40.7	± 2.5
RF+cost aware learning	40.4	54.3	**46.3**	± 2.7
NB+cost aware learning	27.3	**61.5**	37.8	± 2.2

Table 3: Classification scores (%) on the relevant class. CI is the confidence interval on the F-score (RF = Random Forest, NB= naive bayes).

Features	Prec.	Recall	F-score
all	40.4	54.3	46.3
all − corpus feat.	37.4	52.8	43.8
all − similarity feat.	36.1	49.5	41.8
all − contextual feat.	36.5	54.8	43.8

Table 4: Impact of each group of features on the best scores (%) : the lowest the results, the bigger the impact of the removed group of features.

4 Related work

Our work is related to two issues: evaluating distributional resources, and improving them. Evaluating distributional resources is the subject of a lot of methodological reflection (Sahlgren, 2006), and as we said in the introduction, evaluations can be divided between extrinsic and intrinsic evaluations. In extrinsic evaluations, models are evaluated against benchmarks focusing on a single task or a single aspect of a resource: either discriminative, TOEFL-like tests (Freitag et al., 2005), analogy production (Turney, 2008), or synonym selection (Weeds, 2003; Anguiano et al., 2011; Ferret, 2013; Curran and Moens, 2002). In intrinsic evaluations, associations norms are used, such as the 353 word-similarity dataset (Finkelstein et al., 2002), e.g. (Pado and Lapata, 2007; Agirre et al., 2009), or specifically designed test cases, as in (Baroni and Lenci, 2011). We differ from all these evaluation procedures as we do not focus on an *essential* view of the relatedness of two lexical items, but evaluate the link in a context where the relevance of the link is in question, an "existential" view of semantic relatedness.

As for improving distributional thesauri, outside of numerous alternate approaches to the construction, there is a body of work focusing on improving an existing resource, for instance reweighting context features once an initial thesaurus is built (Zhitomirsky-Geffet and Dagan, 2009), or post-processing the resource to filter bad neighbours or re-ranking neighbours of a given target (Ferret, 2013). They still use "essential" evaluation measures (mostly synonym extraction), although the latter comes close to our work since it also trains a model to detect (intrinsically) bad neighbours by using example sentences with the words to discriminate. We are not aware of any work that would try to evaluate differently semantic neighbours according to the context they appear in.

5 Conclusion

We proposed a method to reliably evaluate distributional semantic similarity in a broad sense by considering the validation of lexical pairs in contexts where they both appear. This helps cover non classical semantic relations which are hard to evaluate with classical resources. We also presented a supervised learning model which combines global features from the corpus used to built a distributional thesaurus and local features from the text where similarities are to be judged as relevant or not to the coherence of a document. It seems from these experiments that the quality of distributional relations depends on the contextualizing of the related lexical items, beyond just the simi-

larity score and the ranks of items as neighbours of other items. This can hopefully help filter out lexical pairs when word lexical similarity is used as an information source where context is important: lexical disambiguation (Miller et al., 2012), topic segmentation (Guinaudeau et al., 2012). This can also be a preprocessing step when looking for similarities at higher levels, for instance at the sentence level (Mihalcea et al., 2006) or other macrotextual level (Agirre et al., 2013), since these are always aggregation functions of word similarities. There are limits to what is presented here: we need to evaluate the importance of the level of noise in the distributional neighbours database, or at least the quantity of non-semantic relations present, and this depends on the way the database is built. Our starting corpus is relatively small compared to current efforts in this framework. We are confident that the same methodology can be followed, even though the quantitative results may vary, since it is independent of the particular distributional thesaurus we used, and the way the similarities are computed.

References

E. Agirre, E. Alfonseca, K. Hall, J. Kravalova, M. Paşca, and A. Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *sem 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

E.H. Anguiano, P. Denis, et al. 2011. FreDist: Automatic construction of distributional thesauri for French. In *Actes de la 18eme conférence sur le traitement automatique des langues naturelles*, pages 119–124.

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

M. Baroni and A. Lenci. 2011. How we BLESSed distributional semantic evaluation. *GEMS 2011*, pages 1–10.

Stefan Bordag. 2008. A comparison of co-occurrence and similarity measures as simulations of context. In Alexander F. Gelbukh, editor, *CICLing*, volume 4919 of *Lecture Notes in Computer Science*, pages 52–63. Springer.

D. Bourigault. 2002. UPERY : un outil d'analyse distributionnelle tendue pour la construction d'ontologies partir de corpus. In *Actes de la 9e confrence sur le Traitement Automatique de la Langue Naturelle*, pages 75–84, Nancy.

Leo Breiman. 2001. Random forests. *Machine Learning*, 45(1):5–32.

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. Smote: Synthetic minority over-sampling technique. *J. Artif. Intell. Res. (JAIR)*, 16:321–357.

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

James R. Curran and Marc Moens. 2002. Improvements in automatic thesaurus extraction. In *Proceedings of the ACL-02 Workshop on Unsupervised Lexical Acquisition*, pages 59–66.

J.R. Curran. 2004. *From distributional to semantic similarity*. Ph.D. thesis, University of Edinburgh.

Gaël Dias, Elsa Alves, and José Gabriel Pereira Lopes. 2007. Topic segmentation algorithms for text summarization and passage retrieval: an exhaustive evaluation. In *Proceedings of the 22nd national conference on Artificial intelligence - Volume 2*, AAAI'07, pages 1334–1339. AAAI Press.

Pedro Domingos. 1999. Metacost: A general method for making classifiers cost-sensitive. In Usama M. Fayyad, Surajit Chaudhuri, and David Madigan, editors, *KDD*, pages 155–164. ACM.

Olivier Ferret. 2013. Identifying bad semantic neighbors for improving distributional thesauri. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 561–571, Sofia, Bulgaria, August. Association for Computational Linguistics.

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

Eibe Frank, Mark Hall, , and Len Trigg. 2004. Weka 3.3: Data mining software in java. www.cs.waikato.ac.nz/ml/weka/.

Dayne Freitag, Matthias Blume, John Byrnes, Edmond Chow, Sadik Kapadia, Richard Rohwer, and Zhiqiang Wang. 2005. New experiments in distributional representations of synonymy. In *Proceedings of CoNLL*, pages 25–32, Ann Arbor, Michigan, June. Association for Computational Linguistics.

W. Gale, K. Church, and D. Yarowsky. 1992. One sense per discourse. In *In Proceedings of the 4th DARPA Speech and Natural Language Workshop, New-York*, pages 233–237.

G. Grefenstette. 1994. *Explorations in automatic thesaurus discovery*. Kluwer Academic Pub., Boston.

Camille Guinaudeau, Guillaume Gravier, and Pascale Sébillot. 2012. Enhancing lexical cohesion measure with confidence measures, semantic relations and language model interpolation for multimedia spoken content topic segmentation. *Computer Speech & Language*, 26(2):90–104.

Karen Sparck Jones. 1994. Towards better NLP system evaluation. In *Proceedings of the Human Language Technology Conference*, pages 102–107. Association for Computational Linguistics.

D. Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, pages 296–304, Madison.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the 21st national conference on Artificial intelligence, AAAI06*, volume 1, pages 775–780. AAAI Press.

Tristan Miller, Chris Biemann, Torsten Zesch, and Iryna Gurevych. 2012. Using distributional similarity for lexical expansion in knowledge-based word sense disambiguation. In *Proceedings of COLING 2012*, pages 1781–1796, Mumbai, India, December. The COLING 2012 Organizing Committee.

J. Morris and G. Hirst. 2004. Non-classical lexical semantic relations. In *Proceedings of the HLT Workshop on Computational Lexical Semantics*, pages 46–51, Boston.

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

Thierry Poibeau and Cédric Messiant. 2008. Do we still Need Gold Standards for Evaluation? In *Proceedings of the Language Resource and Evaluation Conference*.

Magnus Sahlgren. 2006. Towards pertinent evaluation methodologies for word-space models. In *In Proceedings of the 5th International Conference on Language Resources and Evaluation*.

G. Salton, C. S. Yang, and C. T. Yu. 1975. A theory of term importance in automatic text analysis. *Journal of the American Society for Information Science*, 26(1):33–44.

Peter D. Turney. 2008. A uniform approach to analogies, synonyms, antonyms, and associations. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 905–912, Stroudsburg, PA, USA. Association for Computational Linguistics.

L. van der Plas. 2008. *Automatic Lexico-Semantic Acquisition for Question Answering*. Ph.D. thesis, University of Groningen.

J. Weeds and D. Weir. 2005. Co-occurrence retrieval: A flexible framework for lexical distributional similarity. *Computational Linguistics*, 31(4):439–475.

Julie Elizabeth Weeds. 2003. *Measures and Applications of Lexical Distributional Similarity*. Ph.D. thesis, University of Sussex.

Maayan Zhitomirsky-Geffet and Ido Dagan. 2009. Bootstrapping distributional feature vector quality. *Computational Linguistics*, 35(3):435–461.

Interpretable Semantic Vectors from a Joint Model of Brain- and Text-Based Meaning

Alona Fyshe[1], **Partha P. Talukdar**[1], **Brian Murphy**[2], **Tom M. Mitchell**[1]
[1]Machine Learning Department, Carnegie Mellon University
[2]School of Electronics, Electrical Engineering and Computer Science
Queen's University Belfast
[afyshe,partha.talukdar,tom.mitchell]@cs.cmu.edu
brian.murphy@qub.ac.uk

Abstract

Vector space models (VSMs) represent word meanings as points in a high dimensional space. VSMs are typically created using a large text corpora, and so represent word semantics as observed in text. We present a new algorithm (JNNSE) that can incorporate a measure of semantics not previously used to create VSMs: brain activation data recorded while people read words. The resulting model takes advantage of the complementary strengths and weaknesses of corpus and brain activation data to give a more complete representation of semantics. Evaluations show that the model 1) matches a behavioral measure of semantics more closely, 2) can be used to predict corpus data for unseen words and 3) has predictive power that generalizes across brain imaging technologies and across subjects. We believe that the model is thus a more faithful representation of mental vocabularies.

1 Introduction

Vector Space Models (VSMs) represent lexical meaning by assigning each word a point in high dimensional space. Beyond their use in NLP applications, they are of interest to cognitive scientists as an objective and data-driven method to discover word meanings (Landauer and Dumais, 1997).

Typically, VSMs are created by collecting word usage statistics from large amounts of text data and applying some dimensionality reduction technique like Singular Value Decomposition (SVD). The basic assumption is that semantics drives a person's language production behavior, and as a result co-occurrence patterns in written text indirectly encode word meaning. The raw co-occurrence statistics are unwieldy, but in the compressed VSM the distance between any two words is conceived to represent their mutual semantic similarity (Sahlgren, 2006; Turney and Pantel, 2010), as perceived and judged by speakers. This space then reflects the "semantic ground truth" of shared lexical meanings in a language community's vocabulary. However corpus-based VSMs have been criticized as being noisy or incomplete representations of meaning (Glenberg and Robertson, 2000). For example, multiple word senses collide in the same vector, and noise from mis-parsed sentences or spam documents can interfere with the final semantic representation.

When a person is reading or writing, the semantic content of each word will be necessarily activated in the mind, and so in patterns of activity over individual neurons. In principle then, brain activity could replace corpus data as input to a VSM, and contemporary imaging techniques allow us to attempt this. Functional Magnetic Resonance Imaging (fMRI) and Magnetoencephalography (MEG) are two brain activation recording technologies that measure neuronal activation in aggregate, and have been shown to have a predictive relationship with models of word meaning (Mitchell et al., 2008; Palatucci et al., 2009; Sudre et al., 2012; Murphy et al., 2012b).[1]

If brain activation data encodes semantics, we theorized that including brain data in a model of semantics could result in a model more consistent with semantic ground truth. However, the inclusion of brain data will only improve a text-based model if brain data contains semantic information not readily available in the corpus. In addition, if a semantic test involves another subject's brain activation data, performance can improve only if the additional semantic information is consistent across brains. Of course, brains differ in shape, size and in connectivity, so additional information encoded in one brain might not translate to an-

[1]For more details on fMRI and MEG, see Section 4.2

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 489–499,
Baltimore, Maryland, USA, June 23-25 2014.

other. Furthermore, different brain imaging technologies measure very different correlates of neuronal activity. Due to these differences, it is possible that one subject's brain activation data cannot improve a model's performance on another subject's brain data, or for brain data collected using a different recording technology. Indeed, inter-subject models of brain activation is an open research area (Conroy et al., 2013), as is learning the relationship between recording technologies (Engell et al., 2012; Hall et al., 2013). Brain data can also be corrupted by many types of noise (e.g. recording room interference, movement artifacts), another possible hindrance to the use of brain data in VSMs.

VSMs are interesting from both engineering and scientific standpoints. In this work we focus on the scientific question: Can the inclusion of brain data improve semantic representations learned from corpus data? What can we learn from such a model? From an engineering perspective, brain activation data will likely never replace text data. Brain activation recordings are both expensive and time consuming to collect, whereas textual data is vast and much of it is free to download. However, from a scientific perspective, combining text and brain data could lead to more consistent semantic models, in turn leading to a better understanding of semantics and semantic modeling generally.

In this paper, we leverage both kinds of data to build a hybrid VSM using a new matrix factorization method (JNNSE). Our hypothesis is that the noise of brain and corpus derived statistics will be largely orthogonal, and so the two data sources will have complementary strengths as input to VSMs. If this hypothesis is correct, we should find that the resulting VSM is more successful in modeling word semantics as encoded in human judgements, as well as separate corpus and brain data that was not used in the derivation of the model. We will show that our method:

1. creates a VSM that is more correlated to an independent measure of word semantics.
2. produces word vectors that are more predictable from the brain activity of different people, even when brain data is collected with a different recording technology.
3. predicts corpus representations of withheld words more accurately than a model that does not combine data sources.
4. directly maps semantic concepts onto the brain by jointly learning neural representations.

Together, these results suggest that corpus and brain activation data measure semantics in compatible and complimentary ways. Our results are evidence that a joint model of brain- and text-based semantics may be closer to semantic ground truth than text-only models. Our findings also indicate that there is additional semantic information available in brain activation data that is not present in corpus data, and that there are elements of semantics currently lacking in text-based VSMs. We have made available the top performing VSMs created with brain and text data (`http://www.cs.cmu.edu/~afyshe/papers/acl2014/`).

In the following sections we will review NNSE, and our extension, JNNSE. We will describe the data used and the experiments to support our position that brain data is a valuable source of semantic information that compliments text data.

2 Non-Negative Sparse Embedding

Non-Negative Sparse Embedding (NNSE) (Murphy et al., 2012a) is an algorithm that produces a latent representation using matrix factorization. Standard NNSE begins with a matrix $X \in \mathbb{R}^{w \times c}$ made of c corpus statistics for w words. NNSE solves the following objective function:

$$\underset{A,D}{\operatorname{argmin}} \sum_{i=1}^{w} \left\| X_{i,:} - A_{i,:} \times D \right\|^2 + \lambda \left\| A \right\|_1 \quad (1)$$

$$\text{subject to: } D_{i,:} D_{i,:}^T \leq 1, \forall\, 1 \leq i \leq \ell \quad (2)$$

$$A_{i,j} \geq 0,\ 1 \leq i \leq w,\ 1 \leq j \leq \ell \quad (3)$$

The solution will find a matrix $A \in \mathbb{R}^{w \times \ell}$ that is sparse, non-negative, and represents word semantics in an ℓ-dimensional latent space. $D \in \mathbb{R}^{\ell \times c}$ gives the encoding of corpus statistics in the latent space. Together, they factor the original corpus statistics matrix X in a way that minimizes the reconstruction error. The L_1 constraint encourages sparsity in A; λ is a hyperparameter. Equation 2 constrains D to eliminate solutions where A is made arbitrarily small by making D arbitrarily large. Equation 3 ensures that A is non-negative. We may increase ℓ to give more dimensional space to represent word semantics, or decrease ℓ for more compact representations.

The sparse and non-negative representation in A produces a more interpretable semantic space, where interpretability is quantified with a behavioral task (Chang et al., 2009; Murphy et al., 2012a). To illustrate the interpretability of NNSE, we describe a word by selecting the word's top scoring dimensions, and selecting the top scoring words in those dimensions. For example, the word chair has the following top scoring dimensions:

1. chairs, seating, couches;
2. mattress, futon, mattresses;
3. supervisor, coordinator, advisor.

These dimensions cover two of the distinct meanings of the word chair (furniture and person of power).

NNSE's sparsity constraint dictates that each word can have a non-zero score in only a few dimensions, which aligns well to previous feature elicitation experiments in psychology. In feature elicitation, participants are asked to name the characteristics (features) of an object. The number of characteristics named is usually small (McRae et al., 2005), which supports the requirement of sparsity in the learned latent space.

3 Joint Non-Negative Sparse Embedding

We extend NNSEs to incorporate an additional source of data for a subset of the words in X, and call the approach Joint Non-Negative Sparse Embeddings (JNNSEs). The JNNSE algorithm is general enough to incorporate any new information about the a word w, but for this study we will focus on brain activation recordings of a human subject reading single words. We will incorporate either fMRI or MEG data, and call the resulting models JNNSE(fMRI+Text) and JNNSE(MEG+Text) and refer to them generally as JNNSE(Brain+Text). For clarity, from here on, we will refer to NNSE as NNSE(Text), or NNSE(Brain) depending on the single source of input data used.

Let us order the rows of the corpus data X so that the first $1 \dots w'$ rows have both corpus statistics and brain activation recordings. Each brain activation recording is a row in the brain data matrix $Y \in \mathbb{R}^{w' \times v}$ where v is the number of features derived from the recording. For MEG recordings, $v =$sensors $\times$ time points$= 306 \times 150$. For fMRI $v =$ grey-matter voxels $=\simeq 20,000$ depending on the brain anatomy of each individual subject. The new objective function is:

$$\underset{A, D^{(c)}, D^{(b)}}{\operatorname{argmin}} \sum_{i=1}^{w} \left\| X_{i,:} - A_{i,:} \times D^{(c)} \right\|^2 + \sum_{i=1}^{w'} \left\| Y_{i,:} - A_{i,:} \times D^{(b)} \right\|^2 + \lambda \left\| A \right\|_1 \quad (4)$$

$$\text{subject to:} \quad D^{(c)}_{i,:} D^{(c)T}_{i,:} \leq 1, \forall\, 1 \leq i \leq \ell \quad (5)$$

$$D^{(b)}_{i,:} D^{(b)T}_{i,:} \leq 1, \forall\, 1 \leq i \leq \ell \quad (6)$$

$$A_{i,j} \geq 0,\ 1 \leq i \leq w,\ 1 \leq j \leq \ell \quad (7)$$

We have introduced an additional constraint on the rows $1 \dots w'$, requiring that some of the learned representations in A also reconstruct the brain activation recordings (Y) through representations in $D^{(b)} \in \mathbb{R}^{\ell \times v}$. Let us use A' to refer to the brain-constrained rows of A. Words that are close in "brain space" must have similar representations in A', which can further percolate to affect the representations of other words in A via closeness in "corpus space".

With A or D fixed, the objective function for NNSE(Text) and JNNSE(Brain+Text) is convex. However, we are solving for A and D, so the problem is non-convex. To solve for this objective, we use the online algorithm of Section 3 from Mairal et al. (Mairal et al., 2010). This algorithm is guaranteed to converge, and in practice we found that JNNSE(Brain+Text) converged as quickly as NNSE(Text) for the same ℓ. We used the SPAMS package[2] to solve, and set $\lambda = 0.025$. This algorithm was a very easy extension to NNSE(Text) and required very little additional tuning.

We also consider learning shared representations in the case where data X and Y contain the effects of known *disjoint* features. For example, when a person reads a word, the recorded brain activation data Y will contain the physiological response to viewing the stimulus, which is unrelated to the semantics of the word. These signals can be attributed to, for example, the number of letters in the word and the number of white pixels on the screen (Sudre et al., 2012). To account for such effects in the data, we augment A' with a set of n fixed, manually defined features (e.g. word length) to create $A'_{percept} \in \mathbb{R}^{w \times (\ell+n)}$. $D^{(b)} \in \mathbb{R}^{(\ell+n) \times v}$ is used with $A'_{percept}$,

[2]SPAMS Package: http://spams-devel.gforge.inria.fr/

to reconstruct the brain data Y. More generally, one could instead allocate a certain number of latent features specific to X *or* Y, both of which could be learned, as explored in some related work (Gupta et al., 2013). We use 11 *perceptual* features that characterize the non-semantic features of the word stimulus (for a list, see supplementary material at `http://www.cs.cmu.edu/~afyshe/papers/acl2014/`).

The JNNSE algorithm is advantageous in that it can handle partially paired data. That is, the algorithm does not require that every row in X also have a row in Y. Fully paired data is a requirement of many other approaches (White et al., 2012; Jia and Darrell, 2010). Our approach allows us to leverage the semantic information in corpus data even for words without brain activation recordings.

JNNSE(Brain+Text) does not require brain data to be mapped to a common average brain, which is often the case when one wants to generalize between human subjects. Such mappings can blur and distort data, making it less useful for subsequent prediction steps. We avoid these mappings, and instead use the fact that similar words elicit similar brain activation *within* a subject. In the JNNSE algorithm, it is this closeness in "brain space" that guides the creation of the latent space A. Leveraging intra-subject distance measures to study inter-subject encodings has been studied previously (Kriegeskorte et al., 2008a; Raizada and Connolly, 2012), and has even been used across species (humans and primates) (Kriegeskorte et al., 2008b).

Though we restrict ourselves to using one subject per JNNSE(Brain+Text) model, the JNNSE algorithm could easily be extended to include data from multiple brain imaging experiments by adding a new squared loss term for additional brain data.

3.1 Related Work

Perhaps the most well known related approach to joining data sources is Canonical Correlation Analysis (CCA) (Hotelling, 1936), which has been applied to brain activation data in the past (Rustandi et al., 2009). CCA seeks two linear transformations that maximally correlate two data sets in the transformed form. CCA requires that the data sources be paired (all rows in the corpus data must have a corresponding brain data), as correlation between points is integral to the objective. To apply CCA to our data we would need to discard the vast majority of our corpus data, and use only the 60 rows of X with corresponding rows in Y. While CCA holds the input data fixed and maximally correlates the transformed form, we hold the transformed form fixed and seek a solution that maximally correlates the reconstruction ($AD^{(c)}$ or $A'D^{(b)}$) with the data (X and Y respectively). This shift in error compensation is what allows our data to be only partially paired. While a Bayesian formulation of CCA can handle missing data, our model has missing data for $> 97\%$ of the full $w \times (v + c)$ brain and corpus data matrix. To our knowledge, this extreme amount of missing data has not been explored with Bayesian CCA.

One could also use a topic model style formulation to represent this semantic representation task. Supervised topic models (Blei and McAuliffe, 2007) use a latent topic to generate two observed outputs: words in a document and a categorical label for the document. The same idea could be applied here: the latent semantic representation generates the observed brain activity and corpus statistics. Generative and discriminative models both have their own strengths and weaknesses, generative models being particularly strong when data sources are limited (Ng and Jordan, 2002). Our task is an interesting blend of data-limited and data-rich problem scenarios.

In the past, various pieces of additional information have been incorporated into semantic models. For example, models with behavioral data (Silberer and Lapata, 2012) and models with visual information (Bruni et al., 2011; Silberer et al., 2013) have both shown to improve semantic representations. Other works have correlated VSMs built with text or images with brain activation data (Murphy et al., 2012b; Anderson et al., 2013). To our knowledge, this work is the first to integrate brain activation data into the construction of the VSM.

4 Data

4.1 Corpus Data

The corpus statistics used here are the downloadable vectors from Fyshe et al. (2013)[3]. They are compiled from a 16 billion word subset of ClueWeb09 (Callan and Hoy, 2009) and contain two types of corpus features: dependency and document features, found to be complimentary for

[3] `http://www.cs.cmu.edu/~afyshe/papers/conll2013/`

most tasks. Dependency statistics were derived by dependency parsing the corpus and compiling counts for all dependencies incident on the word. Document statistics are word-document co-occurrence counts. Count thresholding was applied to reduce noise, and positive pointwise-mutual-information (PPMI) (Church and Hanks, 1990) was applied to the counts. SVD was applied to the document and dependency statistics and the top 1000 dimensions of each type were retained. We selected the rows corresponding to noun-tagged words (approx. 17000 words).

4.2 Brain Activation Data

We have MEG and fMRI data at our disposal. MEG measures the magnetic field caused by many thousands of neurons firing together, and has good time resolution (1000 Hz) but poor spatial resolution. fMRI measures the change in blood oxygenation that results from differential neural activity, and has good spatial resolution but poor time resolution (0.5-1 Hz). We have fMRI data and MEG data for 18 subjects (9 in each imaging modality) viewing 60 concrete nouns (Mitchell et al., 2008; Sudre et al., 2012). The 60 words span 12 word categories (animals, buildings, tools, insects, body parts, furniture, building parts, utensils, vehicles, objects, clothing, food). Each of the 60 words was presented with a line drawing, so word ambiguity is not an issue. For both recording modalities, all trials for a particular word were averaged together to create one training instance per word, with 60 training instances in all for each subject and imaging modality. More preprocessing details appear in the supplementary material.

5 Experimental Results

Here we explore several variations of JNNSE and NNSE formulations. For a comparison of the models used, see Table 1.

5.1 Correlation to Behavioral Data

To test if our joint model of Brain+Text is closer to semantic ground truth we compared the latent representation A learned via JNNSE(Brain+Text) or NNSE(Text) to an independent behavioral measure of semantics. We collected behavioral data for the 60 nouns in the form of answers to 218 semantic questions. Answers were gathered with Mechanical Turk. The full list of questions appear in the supplementary material. Some example questions are:"Is it alive?", and "Can it bend?". Mechanical Turk users were asked to respond to each question for each word on a scale of 1-5. At least 3 respondents answered each question and the median score was used. This gives us a semantic representation of each of the 60 words in a 218-dimensional behavioral space. Because we required answers to each of the questions for all words, we do not have the problems of sparsity that exist for feature production norms from other studies (McRae et al., 2005). In addition, our answers are ratings, rather than binary yes/no answers.

For a given value of ℓ we solve the NNSE(Text) and JNNSE(Brain+Text) objective function as detailed in Equation 1 and 4 respectively. We compared JNNSE(Brain+Text) and NNSE(Text) models by measuring the correlation of all pairwise distances in JNNSE(Brain+Text) and NNSE(Text) space to the pairwise distances in the 218-dimensional semantic space. Distances were calculated using normalized Euclidean distance (equivalent in rank-ordering to cosine distance, but more suitable for sparse vectors). Figure 1 shows the results of this correlation test. The error bars for the JNNSE(Brain+Text) models represent a 95% confidence interval calculated using the standard error of the mean (SEM) over the 9 person-specific JNNSE(Brain+Text) models. Because there is only one NNSE(Text) model for each dimension setting, no SEM can be calculated, but it suffices to show that the NNSE(Text) correlation does not fall into the 95% confidence interval of the JNNSE(Brain+Text) models. The SVD matrix for the original corpus data has correlation 0.4279 to the behavioral data, also below the 95% confidence interval for all JNNSE models. The results show that a model that incorporates brain activation data is more faithful to a behavioral measure of semantics.

5.2 Word Prediction from Brain Activation

We now show that the JNNSE(Brain+Text) vectors are more consistent with independent samples of brain activity collected from different subjects, even when recorded using different recording technologies. As previously mentioned, because there is a large degree of variation between brains and because MEG and fMRI measure very different correlates of neuronal activity, this type of generalization has proven to be very challenging and is an open research question in the neuroscience community.

The output A of the JNNSE(Brain+Text) or

Table 1: A Comparison of the models explored in this paper, and the data upon which they operate.

Model Name	Section(s)	Text Data	Brain Data	Withheld Data
NNSE(Text)	2, 5	✓	x	-
NNSE(Brain)	2, 5.2.1, 5.3	x	✓	-
JNNSE(Brain+Text)	3, 5	✓	✓	-
JNNSE(Brain+Text): Dropout task	5.2.2	✓	✓	subset of brain data
JNNSE(Brain+Text): Predict corpus	5.3	✓	✓	subset of text data

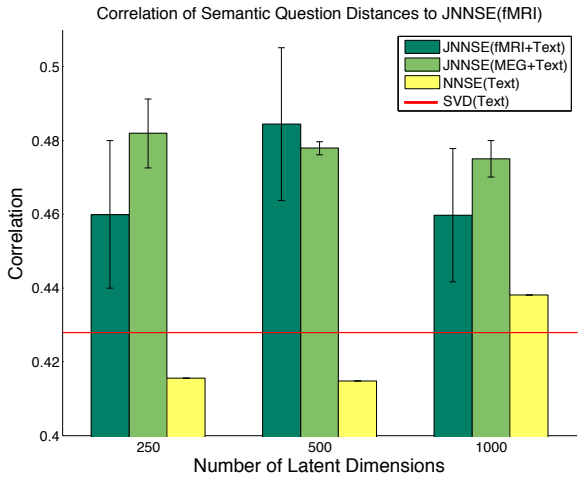


Figure 1: Correlation of JNNSE(Brain+Text) and NNSE(Text) models with the distances in a semantic space constructed from behavioral data. Error bars indicate SEM.

NNSE(Text) algorithm can be used as a VSM, which we use for the task of word prediction from fMRI or MEG recordings. A JNNSE(Brain+Text) created with a particular human subject's data is never used in the prediction framework with that same subject. For example, if we use fMRI data from subject 1 to create a JNNSE(fMRI+Text), we will test it with the remaining 8 fMRI subjects, but all 9 MEG subjects (fMRI and MEG subjects are disjoint).

Let us call the VSM learned with JNNSE(Brain+Text) or NNSE(Text) the *semantic vectors*. We can train a weight matrix W that predicts the semantic vector $\mathbf{a}$ of a word from that word's brain activation vector $\mathbf{x}$: $\mathbf{a} = W\mathbf{x}$. W can be learned with a variety of methods, we will use L_2 regularized regression. One can also train regressors that predict the brain activation data from the semantic vector: $\mathbf{x} = W\mathbf{a}$, but we have found this to give lower predictive accuracy. Note that we must *re-train* our weight matrix W for each subject (instead of re-using $D^{(b)}$ from Equation 4) because testing always occurs on a different subject, and the brain activation data is not inter-subject aligned.

We train ℓ independent L_2 regularized regressors to predict the ℓ-dimensional vectors $\mathbf{a} = \{a_1 \dots a_\ell\}$. The predictions are concatenated to produce a *predicted* semantic vector: $\hat{\mathbf{a}} = \{\hat{a}_1, \dots, \hat{a}_\ell\}$. We assess word prediction performance by testing if the model can differentiate between two unseen words, a task named *2 vs. 2 prediction* (Mitchell et al., 2008; Sudre et al., 2012). We choose the assignment of the two held out semantic vectors $(\mathbf{a}^{(1)}, \mathbf{a}^{(2)})$ to predicted semantic vectors $(\hat{\mathbf{a}}^{(1)}, \hat{\mathbf{a}}^{(2)})$ that minimizes the sum of the two normalized Euclidean distances. 2 vs. 2 accuracy is the percentage of tests where the correct assignment is chosen.

The 60 nouns fall into 12 word categories. Words in the same word category (e.g. screwdriver and hammer) are closer in semantic space than words in different word categories, which makes some 2 vs. 2 tests more difficult than others. We choose 150 random pairs of words (with each word represented equally) to estimate the difficulty of a typical word pair, without having to test all $\binom{60}{2}$ word pairs. The same 150 random pairs are used for all subjects and all VSMs. Expected chance performance on the 2 vs. 2 test is 50%.

Results for testing on fMRI data in the 2 vs. 2 framework appear in Figure 2. JNNSE(fMRI+Text) data performed on average 6% better than the best NNSE(Text), and exceeding even the original SVD corpus representations while maintaining interpretability. These results generalize across brain activity recording types; JNNSE(MEG+Text) performs as well as JNNSE(fMRI+Text) when tested on fMRI data. The results are consistent when testing on MEG data: JNNSE(MEG+Text) or JNNSE(fMRI+Text) outperforms NNSE(Text) (see Figure 3).

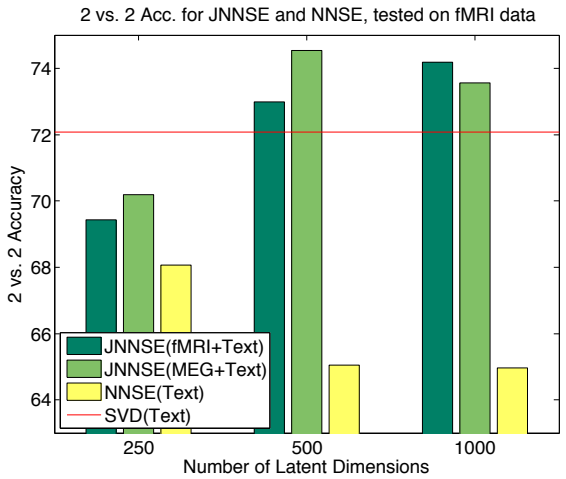


Figure 2: Average 2 vs. 2 accuracy for NNSE(Text) and JNNSE(Brain+Text), tested on fMRI data. Models created with one subject's fMRI data were not used to compute 2 vs. 2 accuracy for that same subject.

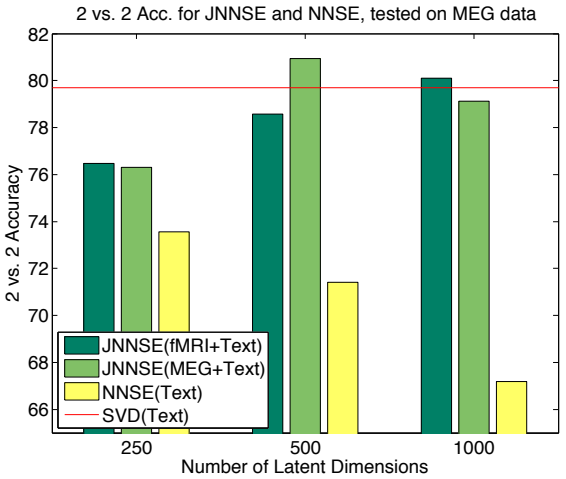


Figure 3: Average 2 vs. 2 accuracy for NNSE(Text) and JNNSE(Brain+Text), tested on MEG data. Models created with one subject's MEG data were not used to compute 2 vs. 2 accuracy for that same subject.

NNSE(Text) performance decreases as the number of latent dimension increases. This implies that without the regularizing effect of brain activation data, the extra NNSE(Text) dimensions are being used to overfit to the corpus data, or possibly to fit semantic properties not detectable with current brain imaging technologies. However, when brain activation data is included, increasing the number of latent dimensions strictly increases performance for JNNSE(fMRI+Text). JNNSE(MEG+Text) has peak performance with 500 latent dimensions, with $\sim$ 1% decrease in performance at 1000 latent dimensions. In previous work, the ability to decode words from brain activation data was found to improve with added latent dimensions (Murphy et al., 2012a). Our results may differ because our words are POS tagged, and we included only nouns for the final NNSE(Text) model. We found that with the original $\lambda = 0.05$ setting from Murphy et al. (Murphy et al., 2012a) produced vectors that were too sparse; four of the 60 test words had all-zero vectors (JNNSE(Brain+Text) models did have any all-zero vectors). To improve the NNSE(Text) vectors for a fair comparison, we reduced $\lambda = 0.025$, under which NNSE(Text) did not produce any all-zero vectors for the 60 words.

Our results show that brain activation data contributes additional information, which leads to an increase in performance for the task of word prediction from brain activation data. This suggests that corpus-only models may not capture all relevant semantic information. This conflicts with previous studies which found that semantic vectors culled from corpus statistics contain all of the semantic information required to predict brain activation (Bullinaria and Levy, 2013).

5.2.1 Prediction from a Brain-only Model

How much predictive power does the corpus data provide to this word prediction task? To test this, we calculated the 2 vs. 2 accuracy for a NNSE(Brain) model trained on brain activation data only. We train NNSE(Brain) with one subject's data and use the resulting vectors to calculate 2 vs. 2 accuracy for the remaining subjects. We have brain data for only 60 words, so using $\ell \geq 60$ latent dimensions leads to an under-constrained system and a degenerate solution wherein only one latent dimension is active for any word (and where the brain data can be perfectly reconstructed). The degenerate solution makes it impossible to generalize across words and leads to performance at chance levels. An NNSE(MEG) trained on MEG data gave maximum 2 vs. 2 accuracy of 67% when $\ell = 20$. The reduced performance may be due to the limited training data and the low SNR of the data, but could also be attributed to the lack of corpus information, which provides another piece of semantic information.

5.2.2 Effect on Rows Without Brain Data

It is possible that some JNNSE(Brain+Text) dimensions are being used exclusively to fit brain activation data, and not the semantics represented in both brain and corpus data. If a particular dimension j is solely used for brain data, the sparsity constraint will favor solutions that sets $A_{(i,j)} = 0$ for $i > w'$ (no brain data constraint), and $A_{(i,j)} > 0$ for some $0 \leq i \leq w'$ (brain data constrained). We found that there were no such dimensions in the JNNSE(Brain+Text). In fact for the $\ell = 1000$ JNNSE(Brain+Text), all latent dimensions had greater than $\sim 25\%$ non-zero entries, which implies that all dimensions are being shared between the two data inputs (corpus and brain activation), and are used to reconstruct both.

To test that the brain activation data is truly influencing rows of A not constrained by brain activation data, we performed a *dropout* test. We split the original 60 words into two 30 word groups (as evenly as possible across word categories). We trained JNNSE(fMRI+Text) with 30 words, and tested word prediction with the remaining 8 subjects and the other 30 words. Thus, the training and testing word sets are disjoint. Because of the reduced size of the training data, we did see a drop in performance, but JNNSE(fMRI+Text) vectors still gave word prediction performance 7% higher than NNSE(Text) vectors. Full results appear in the supplementary material.

5.3 Predicting Corpus Data

Here we ask: can an accurate latent representation of a word be constructed using only brain activation data? This task simulates the scenario where there is no reliable corpus representation of a word, but brain data is available. This scenario may occur for seldom-used words that fall below the thresholds used for the compilation of corpus statistics. It could also be useful for acronym tokens (lol, omg) found in social media contexts where the meaning of the token is actually a full sentence.

We trained a JNNSE(fMRI+Text) with brain data for all 60 words, but withhold the corpus data for 30 of the 60 words (as evenly distributed as possible amongst the 12 word categories). The brain activation data for the 30 withheld words will allow us to create latent representations in A for withheld words. Simultaneously, we will learn a mapping from the latent representation to the corpus data ($D^{(c)}$). This task cannot be performed with a NNSE(Text) model because one cannot learn a latent representation of a word without data of some kind. This further emphasizes the impact of brain imaging data, which will allow us to generalize to previously unseen words in corpus space.

Table 2: Mean rank accuracy over 30 words using corpus representations predicted by a JNNSE(MEG+Text) model trained with some rows of the corpus data withheld. Significance is calculated using Fisher's method to combine p-values for each of the subject-dependent models.

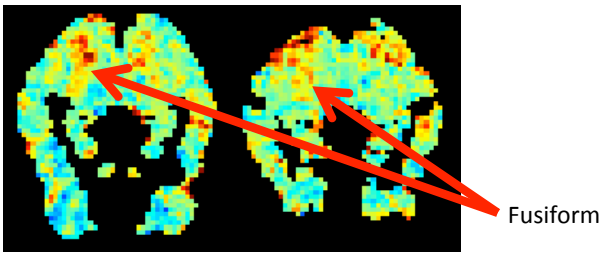
Latent Dim size	Rank Accuracy	p-value
250	65.30	$< 10^{-19}$
500	67.37	$< 10^{-24}$
1000	63.47	$< 10^{-15}$

We use the latent representations in A for each of the words without corpus data and the mapping to corpus space $D^{(c)}$ to predict the withheld corpus data in X. We then rank the withheld rows of X by their distance to the predicted row of X and calculate the mean rank accuracy of the held out words. Results in Table 2 show that we can recreate the withheld corpus data using brain activation data. Peak mean rank accuracy (67.37) is attained at $\ell = 500$ latent dimensions. This result shows that neural semantic representations can create a latent representation that is faithful to unseen corpus statistics, providing further evidence that the two data sources share a strong common element.

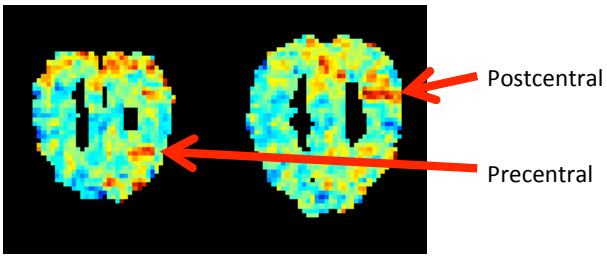
How much power is the remaining corpus data supplying in scenarios where we withhold corpus data? To answer this question, we trained an NNSE(Brain) model on 30 words of brain activation, and then trained a regressor to predict corpus data from those latent brain-only representations. We use the trained regressor to predict the corpus data for the remaining 30 words. Peak performance is attained at $\ell = 10$ latent dimensions, giving mean rank accuracy of 62.37, significantly worse than the model that includes both corpus and brain activation data (67.37).

5.4 Mapping Semantics onto the Brain

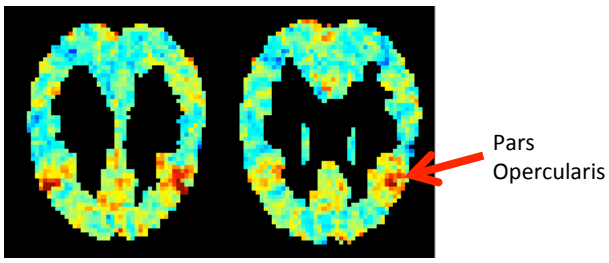
Because our method incorporates brain data into an interpretable semantic model, we can directly map semantic concepts onto the brain. To do this, we examined the mappings from the latent space to the brain space via $D^{(b)}$. We found that the most interpretable mappings come from mod-



(a) $D^{(b)}$ matrix, subject P3, dimension with top words bathroom, balcony, kitchen. MNI coordinates z=-12 (left) and z=-18 (right). Fusiform is associated with shelter words.



(b) $D^{(b)}$ matrix; subject P1; dimension with top words ankle, elbow, knee. MNI coordinates z=60 (left) and z=54 (right). Pre- and post-central areas are activated for body part words.



(c) $D^{(b)}$ matrix; subject P1; dimension with top scoring words buffet, brunch, lunch. MNI coordinates z=30 (left) and z=24 (right). Pars opercularis is believed to be part of the gustatory cortex, which responds to food related words.

Figure 4: The mappings ($D^{(b)}$) from latent semantic space (A) to brain space (Y) for fMRI and words from three semantic categories. Shown are representations of the fMRI slices such that the back of the head is at the top of the image, the front of the head is at the bottom.

els where the perceptual features had been scaled down (divided by a constant factor), which encourages more of the data to be explained by the semantic features in A. Figure 4 shows the mappings ($D^{(b)}$) for dimensions related to shelter, food and body parts. The red areas align with areas of the brain previously known to be activated by the corresponding concepts (Mitchell et al., 2008; Just et al., 2010). Our model has learned these mappings in an unsupervised setting by relating semantic knowledge gleaned from word usage to patterns of activation in the brain. This illustrates how the interpretability of JNNSE can allow one to explore semantics in the human brain. The mappings for one subject are available for download (`http://www.cs.cmu.edu/~afyshe/papers/acl2014/`).

6 Future Work and Conclusion

We are interested in pursuing many future projects inspired by the success of this model. We would like to extend the JNNSE algorithm to incorporate data from multiple subjects, multiple modalities and multiple experiments with non-overlapping words. Including behavioral data and image data is another possibility.

We have explored a model of semantics that incorporates text and brain activation data. Though the number of words for which we have brain activation data is comparatively small, we have shown that including even this small amount of data has a positive impact on the learned latent representations, including for words without brain data. We have provided evidence that the latent representations are closer to the neural representation of semantics, and possibly, closer to semantic ground truth. Our results reveal that there are aspects of semantics not currently represented in text-based VSMs, indicating that there may be room for improvement in either the data or algorithms used to create VSMs. Our findings also indicate that using the brain as a semantic test can separate models that capture this additional semantic information from those that do not. Thus, the brain is an important source of both training and testing data.

Acknowledgments

This work was supported in part by NIH under award 5R01HD075328-02, by DARPA under award FA8750-13-2-0005, and by a fellowship to Alona Fyshe from the Multimodal Neuroimaging Training Program funded by NIH awards T90DA022761 and R90DA023420.

References

Andrew J Anderson, Elia Bruni, Ulisse Bordignon, Massimo Poesio, and Marco Baroni. 2013. Of words , eyes and brains : Correlating image-based distributional semantic models with neural representations of concepts. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing*.

David M Blei and Jon D. McAuliffe. 2007. Supervised topic models. In *Advances in Neural Information Processing Systems*, pages 1–22.

Elia Bruni, Giang Binh Tran, and Marco Baroni. 2011. Distributional semantics from text and images. In *Proceedings of the EMNLP 2011 Geometrical Models for Natural Language Semantics (GEMS)*.

John A Bullinaria and Joseph P Levy. 2013. Limiting factors for mapping corpus-based semantic representations to brain activity. *PloS one*, 8(3):e57191, January.

Jamie Callan and Mark Hoy. 2009. The ClueWeb09 Dataset.

Jonathan Chang, Jordan Boyd-Graber, Sean Gerrish, Chong Wang, and David M Blei. 2009. Reading Tea Leaves : How Humans Interpret Topic Models. In *Advances in Neural Information Processing Systems*, pages 1–9.

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

Bryan R Conroy, Benjamin D Singer, J Swaroop Guntupalli, Peter J Ramadge, and James V Haxby. 2013. Inter-subject alignment of human cortical anatomy using functional connectivity. *NeuroImage*, 81:400–11, November.

Andrew D Engell, Scott Huettel, and Gregory McCarthy. 2012. The fMRI BOLD signal tracks electrophysiological spectral perturbations, not event-related potentials. *NeuroImage*, 59(3):2600–6, February.

Alona Fyshe, Partha Talukdar, Brian Murphy, and Tom Mitchell. 2013. Documents and Dependencies : an Exploration of Vector Space Models for Semantic Composition. In *Computational Natural Language Learning*, Sofia, Bulgaria.

Arthur M Glenberg and David a Robertson. 2000. Symbol Grounding and Meaning: A Comparison of High-Dimensional and Embodied Theories of Meaning. *Journal of Memory and Language*, 43(3):379–401, October.

Sunil Kumar Gupta, Dinh Phung, Brett Adams, and Svetha Venkatesh. 2013. Regularized nonnegative shared subspace learning. *Data Mining and Knowledge Discovery*, 26(1):57–97.

Emma L Hall, Siân E Robson, Peter G Morris, and Matthew J Brookes. 2013. The relationship between MEG and fMRI. *NeuroImage*, November.

Harold Hotelling. 1936. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377.

Yangqing Jia and Trevor Darrell. 2010. Factorized Latent Spaces with Structured Sparsity. In *Advances in Neural Information Processing Systems*, volume 23.

Marcel Adam Just, Vladimir L Cherkassky, Sandesh Aryal, and Tom M Mitchell. 2010. A neurosemantic theory of concrete noun representation based on the underlying brain codes. *PloS one*, 5(1):e8622, January.

Nikolaus Kriegeskorte, Marieke Mur, and Peter Bandettini. 2008a. Representational similarity analysis - connecting the branches of systems neuroscience. *Frontiers in systems neuroscience*, 2(November):4, January.

Nikolaus Kriegeskorte, Marieke Mur, Douglas A Ruff, Roozbeh Kiani, Jerzy Bodurka, Hossein Esteky, Keiji Tanaka, and Peter A Bandettin. 2008b. Matching Categorical Object Representations in Inferior Temporal Cortex of Man and Monkey. *Neuron*, 60(6):1126–1141.

TK Landauer and ST Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 1(2):211–240.

Julien Mairal, Francis Bach, J Ponce, and Guillermo Sapiro. 2010. Online learning for matrix factorization and sparse coding. *The Journal of Machine Learning Research*, 11:19–60.

Ken McRae, George S Cree, Mark S Seidenberg, and Chris McNorgan. 2005. Semantic feature production norms for a large set of living and nonliving things. *Behavior research methods*, 37(4):547–59, November.

Tom M Mitchell, Svetlana V Shinkareva, Andrew Carlson, Kai-Min Chang, Vicente L Malave, Robert A Mason, and Marcel Adam Just. 2008. Predicting human brain activity associated with the meanings of nouns. *Science (New York, N.Y.)*, 320(5880):1191–5, May.

Brian Murphy, Partha Talukdar, and Tom Mitchell. 2012a. Learning Effective and Interpretable Semantic Models using Non-Negative Sparse Embedding. In *Proceedings of Conference on Computational Linguistics (COLING)*.

Brian Murphy, Partha Talukdar, and Tom Mitchell. 2012b. Selecting Corpus-Semantic Models for Neurolinguistic Decoding. In *First Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 114–123, Montreal, Quebec, Canada.

Andrew Y. Ng and Michael I. Jordan. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, volume 14.

Mark Palatucci, Geoffrey Hinton, Dean Pomerleau, and Tom M Mitchell. 2009. Zero-Shot Learning with Semantic Output Codes. *Advances in Neural Information Processing Systems*, 22:1410–1418.

Rajeev D S Raizada and Andrew C Connolly. 2012. What Makes Different People's Representations Alike : Neural Similarity Space Solves the Problem of Across-subject fMRI Decoding. *Journal of Cognitive Neuroscience*, 24(4):868–877.

Indrayana Rustandi, Marcel Adam Just, and Tom M Mitchell. 2009. Integrating Multiple-Study Multiple-Subject fMRI Datasets Using Canonical Correlation Analysis. In *MICCAI 2009 Workshop: Statistical modeling and detection issues in intra- and inter-subject functional MRI data analysis*.

Magnus Sahlgren. 2006. *The Word-Space Model Using distributional analysis to represent syntagmatic and paradigmatic relations between words*. Doctor of philosophy, Stockholm University.

Carina Silberer and Mirella Lapata. 2012. Grounded models of semantic representation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1423–1433.

Carina Silberer, Vittorio Ferrari, and Mirella Lapata. 2013. Models of Semantic Representation with Visual Attributes. In *Association for Computational Linguistics 2013*, Sofia, Bulgaria.

Gustavo Sudre, Dean Pomerleau, Mark Palatucci, Leila Wehbe, Alona Fyshe, Riitta Salmelin, and Tom Mitchell. 2012. Tracking Neural Coding of Perceptual and Semantic Features of Concrete Nouns. *NeuroImage*, 62(1):463–451, May.

Peter D Turney and Patrick Pantel. 2010. From Frequency to Meaning : Vector Space Models of Semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Martha White, Yaoliang Yu, Xinhua Zhang, and Dale Schuurmans. 2012. Convex multi-view subspace learning. In *Advances in Neural Information Processing Systems*, pages 1–14.

Single-Agent vs. Multi-Agent Techniques for Concurrent Reinforcement Learning of Negotiation Dialogue Policies

Kallirroi Georgila, Claire Nelson, David Traum
University of Southern California Institute for Creative Technologies
12015 Waterfront Drive, Playa Vista, CA 90094, USA
{kgeorgila,traum}@ict.usc.edu

Abstract

We use single-agent and multi-agent Reinforcement Learning (RL) for learning dialogue policies in a resource allocation negotiation scenario. Two agents learn concurrently by interacting with each other without any need for simulated users (SUs) to train against or corpora to learn from. In particular, we compare the Q-learning, Policy Hill-Climbing (PHC) and Win or Learn Fast Policy Hill-Climbing (PHC-WoLF) algorithms, varying the scenario complexity (state space size), the number of training episodes, the learning rate, and the exploration rate. Our results show that generally Q-learning fails to converge whereas PHC and PHC-WoLF always converge and perform similarly. We also show that very high gradually decreasing exploration rates are required for convergence. We conclude that multi-agent RL of dialogue policies is a promising alternative to using single-agent RL and SUs or learning directly from corpora.

1 Introduction

The dialogue policy of a dialogue system decides on which actions the system should perform given a particular dialogue state (i.e., dialogue context). Building a dialogue policy can be a challenging task especially for complex applications. For this reason, recently much attention has been drawn to machine learning approaches to dialogue management and in particular Reinforcement Learning (RL) of dialogue policies (Williams and Young, 2007; Rieser et al., 2011; Jurčíček et al., 2012).

Typically there are three main approaches to the problem of learning dialogue policies using RL: (1) learn against a simulated user (SU), i.e., a model that simulates the behavior of a real user (Georgila et al., 2006; Schatzmann et al., 2006); (2) learn directly from a corpus (Henderson et al., 2008; Li et al., 2009); or (3) learn via live interaction with human users (Singh et al., 2002; Gašić et al., 2011; Gašić et al., 2013).

We propose a fourth approach: concurrent learning of the system policy and the SU policy using multi-agent RL techniques. Both agents are trained simultaneously and there is no need for building a SU separately or having access to a corpus.[1] As we discuss below, concurrent learning could potentially be used for learning via live interaction with human users. Moreover, for negotiation in particular there is one more reason in favor of concurrent learning as opposed to learning against a SU. Unlike slot-filling domains, in negotiation the behaviors of the system and the user are symmetric. They are both negotiators, thus building a good SU is as difficult as building a good system policy.

So far research on using RL for dialogue policy learning has focused on single-agent RL techniques. Single-agent RL methods make the assumption that the system learns by interacting with a stationary environment, i.e., an environment that does not change over time. Here the environment is the user. Generally the assumption that users do not significantly change their behavior over time holds for simple information providing tasks (e.g., reserving a flight). But this is not necessarily the case for other genres of dialogue, including negotiation. Imagine a situation where a negotiator is so uncooperative and arrogant that the other negotiators decide to completely change their negotiation strategy in order to punish her. Therefore it is important to investigate RL approaches that do not make such assumptions about the user/environment.

[1] Though corpora or SUs may still be useful for bootstrapping the policies and encoding real user behavior (see section 6).

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 500–510,
Baltimore, Maryland, USA, June 23-25 2014.

Multi-agent RL is designed to work for non-stationary environments. In this case the environment of a learning agent is one or more other agents that can also be learning at the same time. Therefore, unlike single-agent RL, multi-agent RL can handle changes in user behavior or in the behavior of other agents participating in the interaction, and thus potentially lead to more realistic dialogue policies in complex dialogue scenarios. This ability of multi-agent RL can also have important implications for learning via live interaction with human users. Imagine a system that learns to change its strategy as it realizes that a particular user is no longer a novice user, or that a user no longer cares about five star restaurants.

We apply multi-agent RL to a resource allocation negotiation scenario. Two agents with different preferences negotiate about how to share resources. We compare Q-learning (a single-agent RL algorithm) with two multi-agent RL algorithms: Policy Hill-Climbing (PHC) and Win or Learn Fast Policy Hill-Climbing (PHC-WoLF) (Bowling and Veloso, 2002). We vary the scenario complexity (i.e., the quantity of resources to be shared and consequently the state space size), the number of training episodes, the learning rate, and the exploration rate.

Our research contributions are as follows: (1) we propose concurrent learning using multi-agent RL as a way to deal with some of the issues of current approaches to dialogue policy learning (i.e., the need for SUs and corpora), which may also potentially prove useful for learning via live interaction with human users; (2) we show that concurrent learning can address changes in user behavior over time, and requires multi-agent RL techniques and variable exploration rates; (3) to our knowledge this is the first time that PHC and PHC-WoLF are used for learning dialogue policies; (4) for the first time, the above techniques are applied to a negotiation domain; and (5) this is the first study that compares Q-learning, PHC, and PHC-WoLF in such a variety of situations (varying a large number of parameters).

The paper is structured as follows. Section 2 presents related work. Section 3 provides a brief introduction to single-agent RL and multi-agent RL. Section 4 describes our negotiation domain and experimental setup. In section 5 we present our results. Finally, section 6 concludes and provides some ideas for future work.

2 Related Work

Most research in RL for dialogue management has been done in the framework of slot-filling applications such as restaurant recommendations (Lemon et al., 2006; Thomson and Young, 2010; Gašić et al., 2012; Daubigney et al., 2012), flight reservations (Henderson et al., 2008), sightseeing recommendations (Misu et al., 2010), appointment scheduling (Georgila et al., 2010), etc. RL has also been applied to question-answering (Misu et al., 2012), tutoring domains (Tetreault and Litman, 2008; Chi et al., 2011), and learning negotiation dialogue policies (Heeman, 2009; Georgila and Traum, 2011; Georgila, 2013).

As mentioned in section 1, there are three main approaches to the problem of learning dialogue policies using RL.

In the first approach, a SU is hand-crafted or learned from a small corpus of human-human or human-machine dialogues. Then the dialogue policy can be learned by having the system interact with the SU for a large number of dialogues (usually thousands of dialogues). Depending on the application, building a realistic SU can be just as difficult as building a good dialogue policy. Furthermore, it is not clear what constitutes a good SU for dialogue policy learning. Should the SU resemble real user behavior as closely as possible, or should it exhibit some degree of randomness to explore a variety of interaction patterns? Despite much research on the issue, these are still open questions (Schatzmann et al., 2006; Ai and Litman, 2008; Pietquin and Hastie, 2013).

In the second approach, no SUs are required. Instead the dialogue policy is learned directly from a corpus of human-human or human-machine dialogues. For example, Henderson et al. (2008) used a combination of RL and supervised learning to learn a dialogue policy in a flight reservation domain, whereas Li et al. (2009) used Least-Squares Policy Iteration (Lagoudakis and Parr, 2003), an RL-based technique that can learn directly from corpora, in a voice dialer application. However, collecting such corpora is not trivial, especially in new domains. Typically, data are collected in a Wizard-of-Oz setup where human users think that they interact with a system while in fact they interact with a human pretending to be the system, or by having human users interact with a preliminary version of the dialogue system. In both cases the resulting interactions are expected to be quite dif-

ferent from the interactions of human users with the final system. In practice this means that dialogue policies learned from such data could be far from optimal.

The first experiment on learning via live interaction with human users (third approach) was reported by Singh et al. (2002). They used RL to help the system with two choices: how much initiative it should allow the user, and whether or not to confirm information provided by the user. Recently, learning of "full" dialogue policies (not just choices at specific points in the dialogue) via live interaction with human users has become possible with the use of Gaussian processes (Engel et al., 2005; Rasmussen and Williams, 2006). Typically learning a dialogue policy is a slow process requiring thousands of dialogues, hence the need for SUs. Gaussian processes have been shown to speed up learning. This fact together with easy access to a large number of human users through crowd-sourcing has allowed dialogue policy learning via live interaction with human users (Gašić et al., 2011; Gašić et al., 2013).

Space constraints prevent us from providing an exhaustive list of previous work on using RL for dialogue management. Thus below we focus only on research that is directly related to our work, specifically research on concurrent learning of the policies of multiple agents, and the application of RL to negotiation domains.

So far research on RL in the dialogue community has focused on using single-agent RL techniques where the stationary environment is the user. Most approaches assume that the user goal is fixed and that the behavior of the user is rational. Other approaches account for changes in user goals (Ma, 2013). In either case, one can build a user simulation model that is the average of different user behaviors or learn a policy from a corpus that contains a variety of interaction patterns, and thus safely assume that single-agent RL techniques will work. However, in the latter case if the behavior of the user changes significantly over time then the assumption that the environment is stationary will no longer hold.

There has been a lot of research on multi-agent RL in the optimal control and robotics communities (Littman, 1994; Hu and Wellman, 1998; Busoniu et al., 2008). Here two or more agents learn simultaneously. Thus the environment of an agent is one or more other agents that continuously change their behavior because they are also learning at the same time. Therefore the environment is no longer stationary and single-agent RL techniques do not work well or do not work at all. We are particularly interested in the work of Bowling and Veloso (2002) who proposed the PHC and PHC-WoLF algorithms that we use in this paper. We chose these two algorithms because, unlike other multi-agent RL methods (Littman, 1994; Hu and Wellman, 1998), they do not make assumptions that do not always hold and do not require quadratic or linear programming that does not always scale.

English and Heeman (2005) were the first in the dialogue community to explore the idea of concurrent learning of dialogue policies. However, English and Heeman (2005) did not use multi-agent RL but only standard single-agent RL, in particular an on-policy Monte Carlo method (Sutton and Barto, 1998). But single-agent RL techniques are not well suited for concurrent learning where each agent is trained against a continuously changing environment. Indeed, English and Heeman (2005) reported problems with convergence. Chandramohan et al. (2012) proposed a framework for co-adaptation of the dialogue policy and the SU using single-agent RL. They applied Inverse Reinforcement Learning (IRL) (Abbeel and Ng, 2004) to a corpus in order to learn the reward functions of both the system and the SU. Furthermore, Cuayáhuitl and Dethlefs (2012) used hierarchical multi-agent RL for co-ordinating the verbal and non-verbal actions of a robot. Cuayáhuitl and Dethlefs (2012) did not use PHC or PHC-WoLF and did not compare against single-agent RL methods.

With regard to using RL for learning negotiation policies, the amount of research that has been performed is very limited compared to slot-filling. English and Heeman (2005) learned negotiation policies for a furniture layout task. Then Heeman (2009) extended this work by experimenting with different representations of the RL state in the same domain (this time learning against a hand-crafted SU). In both cases, to reduce the search space, the RL state included only information about e.g., whether there was a pending proposal rather than the actual value of this proposal. Paruchuri et al. (2009) performed a theoretical study on how Partially Observable Markov Decision Processes (POMDPs) can be applied to negotiation domains.

Georgila and Traum (2011) built argumentation dialogue policies for negotiation against users of different cultural norms in a one-issue negotiation scenario. To learn these policies they trained SUs on a spoken dialogue corpus in a florist-grocer negotiation domain, and then tweaked these SUs towards a particular cultural norm using hand-crafted rules. Georgila (2013) learned argumentation dialogue policies from a simulated corpus in a two-issue negotiation scenario (organizing a party). Finally, Nouri et al. (2012) used IRL to learn a model for cultural decision-making in a simple negotiation game (the Ultimatum Game).

3 Single-Agent vs. Multi-Agent Reinforcement Learning

Reinforcement Learning (RL) is a machine learning technique used to learn the policy of an agent, i.e., which action the agent should perform given its current state (Sutton and Barto, 1998). The goal of an RL-based agent is to maximize the reward it gets during an interaction. Because it is very difficult for the agent to know what will happen in the rest of the interaction, the agent must select an action based on the average reward it has previously observed after having performed that action in similar contexts. This average reward is called *expected future reward*. Single-agent RL is used in the framework of Markov Decision Processes (MDPs) (Sutton and Barto, 1998) or Partially Observable Markov Decision Processes (POMDPs) (Williams and Young, 2007). Here we focus on MDPs.

An MDP is defined as a tuple (S, A, T, R, γ) where S is the set of states (representing different contexts) which the agent may be in, A is the set of actions of the agent, T is the transition function $S \times A \times S \rightarrow$ [0, 1] which defines a set of transition probabilities between states after taking an action, R is the reward function $S \times A \rightarrow \Re$ which defines the reward received when taking an action from the given state, and γ is a factor that discounts future rewards. Solving the MDP means finding a policy $\pi : S \rightarrow A$. The quality of the policy π is measured by the expected discounted (with discount factor γ) future reward also called Q-value, $Q^{\pi} : S \times A \rightarrow \Re$.

A *stochastic game* is defined as a tuple (n, S, $A_{1...n}$, T, $R_{1...n}$, γ) where n is the number of agents, S is the set of states, A_i is the set of actions available for agent i (and A is the joint action space $A_1 \times A_2 \times ... \times A_n$), T is the transition function $S \times A \times S \rightarrow$ [0, 1] which defines a set of transition probabilities between states after taking a joint action, R_i is the reward function for the ith agent $S \times A \rightarrow \Re$, and γ is a factor that discounts future rewards. The goal is for each agent i to learn a mixed policy $\pi_i : S \times A_i \rightarrow$ [0, 1] that maps states to mixed strategies, which are probability distributions over the agent's actions, so that the agent's expected discounted (with discount factor γ) future reward is maximized.

Stochastic games are a generalization of MDPs for multi-agent RL. In stochastic games there are many agents that select actions and the next state and rewards depend on the joint action of all these agents. The agents can have different reward functions. Partially Observable Stochastic Games (POSGs) are the equivalent of POMDPs for multi-agent RL. In POSGs, the agents have different observations, and uncertainty about the state they are in and the beliefs of their interlocutors. POSGs are very hard to solve but new algorithms continuously emerge in the literature.

In this paper we use three algorithms: Q-learning, Policy Hill-Climbing (PHC), and Win or Learn Fast Policy Hill-Climbing (PHC-WoLF). PHC is an extension of Q-learning. For all three algorithms, Q-values are updated as follows:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a)+\alpha\left(r+\gamma \text{max}_{a'}Q(s^{'},a^{'})\right) \quad (1)$$

In Q-learning, for a given state s, the agent performs the action with the highest Q-value for that state. In addition to Q-values, PHC and PHC-WoLF also maintain the current mixed policy $\pi(s,a)$. In each step the mixed policy is updated by increasing the probability of selecting the highest valued action according to a learning rate δ (see equations (2), (3), and (4) below).

$$\pi(s,a) \leftarrow \pi(s,a) + \Delta_{sa} \quad (2)$$

$$\Delta_{sa} = \begin{cases} -\delta_{sa} & \text{if } a \neq \text{argmax}_{a'}Q(s,a^{'}) \\ \Sigma_{a' \neq a}\delta_{sa'} & \text{otherwise} \end{cases} \quad (3)$$

$$\delta_{sa} = \min\left(\pi(s,a), \frac{\delta}{|A_i|-1}\right) \quad (4)$$

The difference between PHC and PHC-WoLF is that PHC uses a constant learning rate δ whereas

PHC-WoLF uses a variable learning rate (see equation (5) below). The main idea is that when the agent is "winning" the learning rate δ_W should be low so that the opponents have more time to adapt to the agent's policy, which helps with convergence. On the other hand when the agent is "losing" the learning rate δ_{LF} should be high so that the agent has more time to adapt to the other agents' policies, which also facilitates convergence. Thus PHC-WoLF uses two learning rates δ_W and δ_{LF}. PHC-WoLF determines whether the agent is "winning" or "losing" by comparing the current policy's $\pi(s,a)$ expected payoff with that of the average policy $\tilde{\pi}(s,a)$ over time. If the current policy's expected payoff is greater then the agent is "winning", otherwise it is "losing".

$$\delta = \begin{cases} \delta_W & \text{if } \begin{cases} \Sigma_{\alpha'} \pi(s,\alpha') Q(s,\alpha') > \\ \Sigma_{\alpha'} \tilde{\pi}(s,\alpha') Q(s,\alpha') \end{cases} \\ \delta_{LF} & \text{otherwise} \end{cases} \quad (5)$$

More details about Q-learning, PHC, and PHC-WoLF can be found in (Sutton and Barto, 1998; Bowling and Veloso, 2002).

As discussed in sections 1 and 2, single-agent RL techniques, such as Q-learning, are not suitable for multi-agent RL. Nevertheless, despite its shortcomings Q-learning has been used successfully for multi-agent RL (Claus and Boutilier, 1998). Indeed, as we see in section 5, Q-learning can converge to the optimal policy for small state spaces. However, as the state space size increases the performance of Q-learning drops (compared to PHC and PHC-WoLF).

4 Domain and Experimental Setup

Our domain is a resource allocation negotiation scenario. Two agents negotiate about how to share resources. For the sake of readability from now on we will refer to apples and oranges.

The two agents have different goals. Also, they have human-like constraints of imperfect information about each other; they do not know each other's reward function or degree of rationality (during learning our agents can be irrational). Thus a Nash equilibrium (if there exists one) cannot be computed in advance. Agent 1 cares more about apples and Agent 2 cares more about oranges. Table 1 shows the points that Agents 1 and 2 earn for each apple and each orange that they have at the end of the negotiation.

	Agent 1	**Agent 2**
apple	300	200
orange	200	300

Table 1: Points earned by Agents 1 and 2 for each apple and each orange that they have at the end of the negotiation.

Agent 1: offer-2-2 (I offer you 2 A and 2 O)
Agent 2: offer-3-0 (I offer you 3 A and 0 O)
Agent 1: offer-0-3 (I offer you 0 A and 3 O)
Agent 2: offer-4-0 (I offer you 4 A and 0 O)
Agent 1: accept (I accept your offer)

Figure 1: Example interaction between Agents 1 and 2 (A: apples, O: oranges).

We use a simplified dialogue model with two types of speech acts: offers and acceptances. The dialogue proceeds as follows: one agent makes an offer, e.g., "I give you 3 apples and 1 orange", and the other agent may choose to accept it or make a new offer. The negotiation finishes when one of the agents accepts the other agent's offer or time runs out.

We compare Q-learning with PHC and PHC-WoLF. For all algorithms and experiments each agent is rewarded only at the end of the dialogue based on the negotiation outcome (see Table 1). Thus the two agents have different reward functions. There is also a penalty of -10 for each agent action to ensure that dialogues are not too long. Also, to avoid long dialogues, if none of the agents accepts the other agent's offers, the negotiation finishes after 20 pairs of exchanges between the two agents (20 offers from Agent 1 and 20 offers from Agent 2).

An example interaction between the two agents is shown in Figure 1. As we can see, each agent can offer any combination of apples and oranges. So if we have X apples and Y oranges for sharing, there can be $(X+1) \times (Y+1)$ possible offers. For example if we have 2 apples and 2 oranges for sharing, there can be 9 possible offers: "offer-0-0", "offer-0-1", ..., "offer-2-2". For our experiments we vary the number of fruits to be shared and choose to keep X equal to Y.

Table 2 shows our state representation, i.e., the state variables that we keep track of with all the possible values they can take, where X is the num-

Current offer: $(X + 1) \times (Y + 1)$ possible values
How many times the current offer has already been rejected: (0, 1, 2, 3, or 4)
Is the current offer accepted: yes, no

Table 2: State variables.

ber of apples and Y is the number of oranges to be shared. The third variable is always set to "no" until one of the agents accepts the other agent's offer.

Table 3 shows the state and action space sizes for different numbers of apples and oranges to be shared used in our experiments below. The number of actions includes the acceptance of an offer. Table 3 also shows the number of state-action pairs (Q-values). As we will see in section 5, even though the number of states for each agent is not large, it takes many iterations and high exploration rates for convergence due to the fact that both agents are learning at the same time and the assumption of interacting with a stationary environment no longer holds. For comparison, in (English and Heeman, 2005) the state specification for each agent included 5 binary variables resulting in 32 possible states. English and Heeman (2005) kept track of whether there was an offer on the table but not of the actual value of the offer. For our task it is essential to keep track of the offer values, which of course results in much larger state spaces. Also, in (English and Heeman, 2005) there were 5 possible actions resulting in 160 state-action pairs. Our state and action spaces are much larger and furthermore we explore the effect of different state and action space sizes on convergence.

During learning the two agents interact for 5 epochs. Each epoch contains N number of episodes. We vary N from 25,000 up to 400,000 with a step of 25,000 episodes. English and Heeman (2005) trained their agents for 200 epochs, where each epoch contained 200 episodes.

We also vary the exploration rate per epoch. In particular, in the experiments reported in section 5.1 the exploration rate is set as follows: 0.95 for epoch 1, 0.8 for epoch 2, 0.5 for epoch 3, 0.3 for epoch 4, and 0.1 for epoch 5. Section 5.2 reports results again with 5 epochs of training but a constant exploration rate per epoch set to 0.3. An exploration rate of 0.3 means that 30% of the time the agent will select an action randomly.

Finally, we vary the learning rate. For PHC-WoLF we set δ_W = 0.05 and δ_{LF} = 0.2 (see section 3). These values were chosen with experimentation and the basic idea is that the agent should learn faster when "losing" and slower when "winning". For PHC we explore two cases. In the first case which from now on will be referred to as PHC-W, we set δ to be equal to δ_W (also used for PHC-WoLF). In the second case which from now on will be referred to as PHC-LF, we set δ to be equal to δ_{LF} (also used for PHC-WoLF). So unlike PHC-WoLF, PHC-W and PHC-LF do not use a variable learning rate. PHC-W always learns slowly and PHC-LF always learns fast.

	#States	**#Actions**	**#State-Action Pairs**
1 A & O	40	5	200
2 A & O	90	10	900
3 A & O	160	17	2720
4 A & O	250	26	6500
5 A & O	360	37	13320
6 A & O	490	50	24500
7 A & O	640	65	41600

Table 3: State space, action space, and state-action space sizes for different numbers of apples and oranges to be shared (A: apples, O: oranges).

In all the above cases, training stops after 5 epochs. Then we test the learned policies against each other for one more epoch the size of which is the same as the size of the epochs used for training. For example, if the policies were learned for 5 epochs with each epoch containing 25,000 episodes, then for testing the two policies will interact for another 25,000 episodes. For comparison, English and Heeman (2005) had their agents interact for 5,000 dialogues during testing. To ensure that the policies do not converge by chance, we run the training and test sessions 20 times each and we report averages. Thus all results presented in section 5 are averages of 20 runs.

5 Results

Given that Agent 1 is more interested in apples and Agent 2 cares more about oranges, the maximum total utility solution would be the case where each agent offers to get all the fruits it cares about and to give its interlocutor all the fruits it does not care about, and the other agent accepts this offer. Thus, when converging to the maximum total utility solution, in the case of 4 fruits (4 ap-

ples and 4 oranges), the average reward of the two agents should be 1200 minus 10 for making or accepting an offer. For 5 fruits the average reward should be 1500 minus 10, and so forth. We call 1200 (or 1500) the *convergence reward*, i.e., the reward after converging to the maximum total utility solution if we do not take into account the action penalty. For example, in the case of 4 fruits, if Agent 1 starts the negotiation, after converging to the maximum total utility solution the optimal interaction should be: Agent 1 makes an offer to Agent 2, namely 0 apples and 4 oranges, and Agent 2 accepts. Thus the reward for Agent 1 is 1190, the reward for Agent 2 is 1190, and the average reward of the two agents is also 1190. Also, the convergence reward for Agent 1 is 1200 and the convergence reward for Agent 2 is also 1200.

Below, in all the graphs that we provide, we show the average distance from the convergence reward. This is to make all graphs comparable because in all cases the optimal average distance from the convergence reward of the two agents should be equal to 10 (make the optimal offer or accept the optimal offer that the other agent makes). The formulas for calculating the average distance from the convergence reward are:

$$AD_1 = \frac{\sum_{j=1}^{n_r} |CR_1 - R_{1j}|}{n_r} \quad (6)$$

$$AD_2 = \frac{\sum_{j=1}^{n_r} |CR_2 - R_{2j}|}{n_r} \quad (7)$$

$$AD = \frac{AD_1 + AD_2}{2} \quad (8)$$

where CR_1 is the convergence reward for Agent 1, R_{1j} is the reward of Agent 1 for run j, CR_2 is the convergence reward for Agent 2, and R_{2j} is the reward of Agent 2 for run j. Moreover, AD_1 is the average distance from the convergence reward for Agent 1, AD_2 is the average distance from the convergence reward for Agent 2, and AD is the average of AD_1 and AD_2. All graphs of section 5 show AD values. Also, n_r is the number of runs (in our case always equal to 20). Thus in the case of 4 fruits, we will have CR_1=CR_2=1200, and if for all runs R_{1j}=R_{2j}=1190, then AD=10.

5.1 Variable Exploration Rate

In this section we report results with different exploration rates per training epoch (see section 4).

	Q-learning	**PHC-LF**	**PHC-W**	**PHC-WoLF**
1 A & O	10.5	10	10	10
2 A & O	10.3	10.3	10	10
3 A & O	11.7	10	10	10
4 A & O	15	11.8	11.7	11.7
5 A & O	45.4	29.5	26.5	22.9
6 A & O	60.8	33.4	46.1	33.9
7 A & O	95	56	187.8	88.6

Table 4: Average distance from convergence reward over 20 runs for 100,000 episodes per epoch and for different numbers of fruits to be shared (A: apples, O: oranges). The best possible value is 10.

Table 4 shows the average distance from the convergence reward over 20 runs for 100,000 episodes per epoch, for different numbers of fruits, and for all four methods (Q-learning, PHC-LF, PHC-W, and PHC-WoLF). It is clear that as the state space becomes larger 100,000 training episodes per epoch are not enough for convergence. Also, for 1, 2, and 3 fruits all algorithms converge and perform comparably. As the number of fruits increases, Q-learning starts performing worse than the multi-agent RL algorithms. For 7 fruits PHC-W appears to perform worse than Q-learning but this is because, as we can see in Figure 5, in this case more than 400,000 episodes per epoch are required for convergence. Thus after only 100,000 episodes per epoch all policies still behave somewhat randomly.

Figures 2, 3, 4, and 5 show the average distance from the convergence reward as a function of the number of episodes per epoch during training, for 4, 5, 6, and 7 fruits respectively. For 4 fruits it takes about 125,000 episodes per epoch and for 5 fruits it takes about 225,000 episodes per epoch for the policies to converge. This number rises to approximately 350,000 for 6 fruits and becomes even higher for 7 fruits. Q-learning consistently performs worse than the rest of the algorithms. The differences between PHC-LF, PHC-W, and PHC-WoLF are insignificant, which is a bit surprising given that Bowling and Veloso (2002) showed that PHC-WoLF performed better than PHC in a series of benchmark tasks. In Figures 2 and 3, PHC-LF appears to be reaching convergence slightly faster than PHC-W and PHC-WoLF but this is not statistically significant.

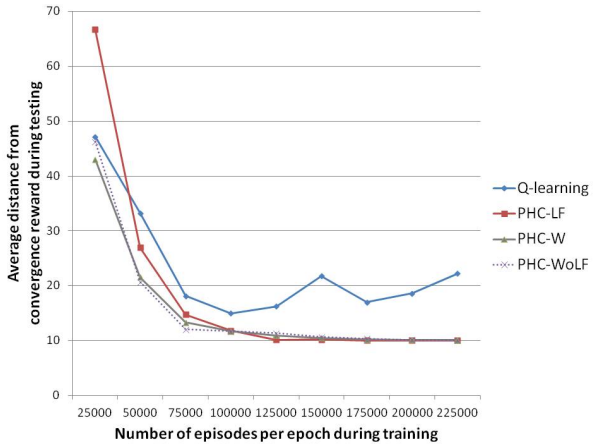


Figure 2: 4 fruits and variable exploration rate: Average distance from convergence reward during testing (20 runs). The best possible value is 10.

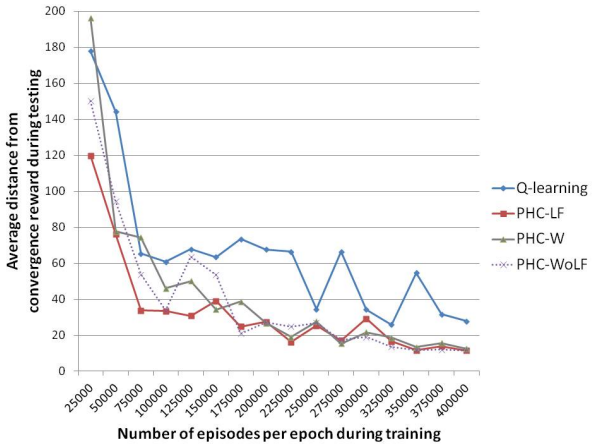


Figure 4: 6 fruits and variable exploration rate: Average distance from convergence reward during testing (20 runs). The best possible value is 10.

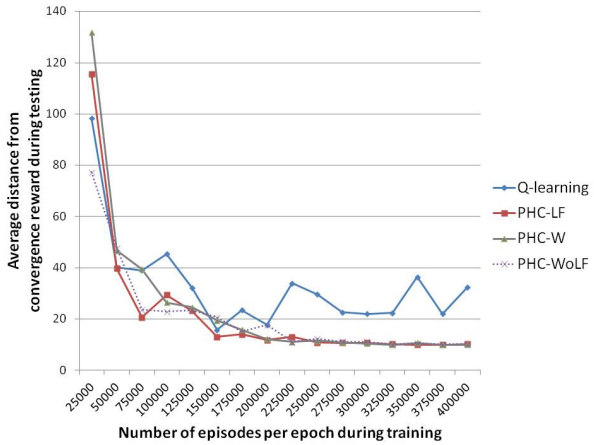


Figure 3: 5 fruits and variable exploration rate: Average distance from convergence reward during testing (20 runs). The best possible value is 10.

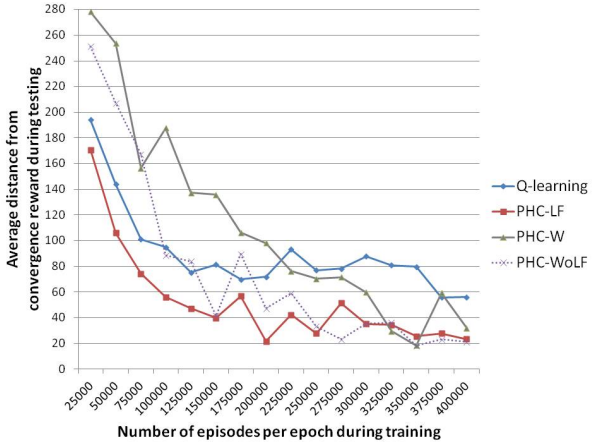


Figure 5: 7 fruits and variable exploration rate: Average distance from convergence reward during testing (20 runs). The best possible value is 10.

5.2 Constant Exploration Rate

In this section we report results with a constant exploration rate for all training epochs (see section 4). Figures 6 and 7 show the average distance from the convergence reward as a function of the number of episodes per epoch during training, for 4 and 5 fruits respectively. Clearly having a constant exploration rate in all epochs is problematic. For 4 fruits, after 225,000 episodes per epoch there is still no convergence. For comparison, with a variable exploration rate it took about 125,000 episodes per epoch for the policies to converge. Likewise for 5 fruits. After 400,000 episodes per epoch there is still no convergence. For comparison, with a variable exploration rate it took about 225,000 episodes per epoch for convergence.

The above results show that, unlike single-agent RL where having a constant exploration rate is perfectly acceptable, here a constant exploration rate does not work.

6 Conclusion and Future Work

We used single-agent RL and multi-agent RL for learning dialogue policies in a resource allocation negotiation scenario. Two agents interacted with each other and both learned at the same time. The advantage of this approach is that it does not require SUs to train against or corpora to learn from.

We compared a traditional single-agent RL algorithm (Q-learning) against two multi-agent RL algorithms (PHC and PHC-WoLF) varying the scenario complexity (state space size), the number

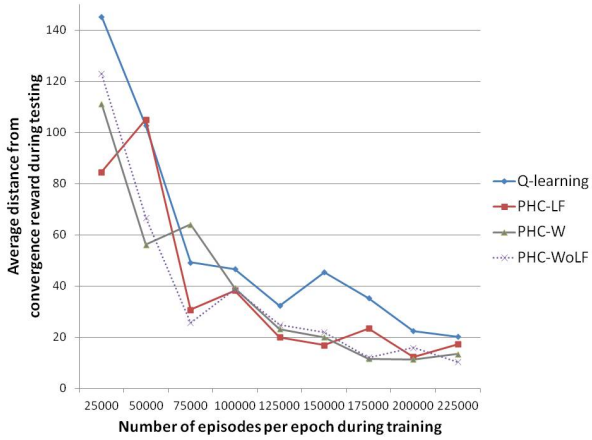


Figure 6: 4 fruits and constant exploration rate: Average distance from convergence reward during testing (20 runs). The best possible value is 10.

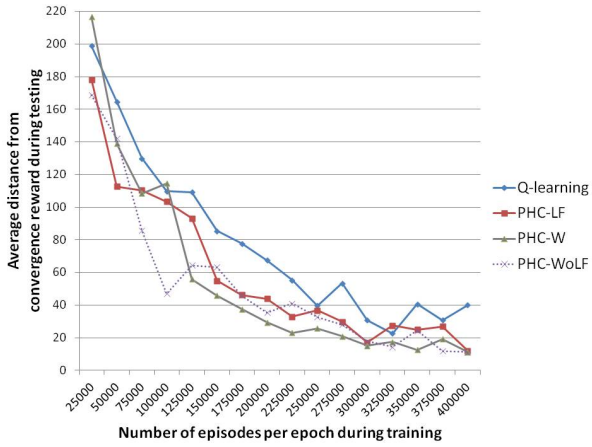


Figure 7: 5 fruits and constant exploration rate: Average distance from convergence reward during testing (20 runs). The best possible value is 10.

of training episodes, and the learning and exploration rates. Our results showed that Q-learning is not suitable for concurrent learning given that it is designed for learning against a stationary environment. Q-learning failed to converge in all cases, except for very small state space sizes. On the other hand, both PHC and PHC-WoLF always converged (or in the case of 7 fruits they needed more training episodes) and performed similarly. We also showed that in concurrent learning very high gradually decreasing exploration rates are required for convergence. We conclude that multi-agent RL of dialogue policies is a promising alternative to using single-agent RL and SUs or learning directly from corpora.

The focus of this paper is on comparing single-agent RL and multi-agent RL for concurrent learning, and studying the implications for convergence and exploration/learning rates. Our next step is testing with human users. We are particularly interested in users whose behavior changes during the interaction and continuous testing against expert repeat users, which has never been done before. Another interesting question is whether corpora or SUs may still be required for designing the state and action spaces and the reward functions of the interlocutors, bootstrapping the policies, and ensuring that information about the behavior of human users is encoded in the resulting learned policies. Gašić et al. (2013) showed that it is possible to learn "full" dialogue policies just via interaction with human users (without any bootstrapping using corpora or SUs). Similarly, concurrent learning could be used in an on-line fashion via live interaction with human users. Or alternatively concurrent learning could be used off-line to bootstrap the policies and then these policies could be improved via live interaction with human users (again using concurrent learning to address possible changes in user behavior). These are open research questions for future work.

Furthermore, we intend to apply multi-agent RL to more complex negotiation domains, e.g., experiment with more than two types of resources (not just apples and oranges) and more types of actions (not just offers and acceptances). We would also like to compare policies learned with multi-agent RL techniques with policies learned with SUs or from corpora both in simulation and with human users. Finally, we aim to experiment with different feature-based representations of the state and action spaces. Currently all possible deal combinations are listed as possible actions and as elements of the state, which can quickly lead to very large state and action spaces as the application becomes more complex (in our case as the number of fruits increases). However, abstraction is not trivial because the agents have no guarantee that the value of a deal is a simple function of the value of its parts, and values may differ for different agents.

Acknowledgments

Claire Nelson sadly died in May 2013. We continued and completed this work after her passing away. She is greatly missed. This work was funded by the NSF grant #1117313.

References

Pieter Abbeel and Andrew Y. Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proc. of the International Conference on Machine Learning*, Bannf, Alberta, Canada.

Hua Ai and Diane Litman. 2008. Assessing dialog system user simulation evaluation measures using human judges. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, Columbus, Ohio, USA.

Michael Bowling and Manuela Veloso. 2002. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250.

L. Busoniu, R. Babuska, and B. De Schutter. 2008. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 38(2):156–172.

Senthilkumar Chandramohan, Matthieu Geist, Fabrice Lefèvre, and Olivier Pietquin. 2012. Co-adaptation in spoken dialogue systems. In *Proc. of the International Workshop on Spoken Dialogue Systems*, Paris, France.

Min Chi, Kurt VanLehn, Diane Litman, and Pamela Jordan. 2011. Empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies. *User Modeling and User-Adapted Interaction*, 21(1-2):137–180.

Caroline Claus and Craig Boutilier. 1998. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proc. of the National Conference on Artificial Intelligence*.

Heriberto Cuayáhuitl and Nina Dethlefs. 2012. Hierarchical multiagent reinforcement learning for coordinating verbal and nonverbal actions in robots. In *Proc. of the ECAI Workshop on Machine Learning for Interactive Systems*, Montpellier, France.

Lucie Daubigney, Matthieu Geist, Senthilkumar Chandramohan, and Olivier Pietquin. 2012. A comprehensive reinforcement learning framework for dialogue management optimization. *IEEE Journal of Selected Topics in Signal Processing*, 6(8):891–902.

Yaakov Engel, Shie Mannor, and Ron Meir. 2005. Reinforcement learning with Gaussian processes. In *Proc. of the International Conference on Machine Learning*, Bonn, Germany.

Michael S. English and Peter A. Heeman. 2005. Learning mixed initiative dialogue strategies by using reinforcement learning on both conversants. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, Vancouver, Canada.

M. Gašić, Filip Jurčíček, Blaise Thomson, Kai Yu, and Steve Young. 2011. On-line policy optimisation of spoken dialogue systems via live interaction with human subjects. In *Proc. of the IEEE Automatic Speech Recognition and Understanding Workshop*, Big Island, Hawaii, USA.

Milica Gašić, Matthew Henderson, Blaise Thomson, Pirros Tsiakoulis, and Steve Young. 2012. Policy optimisation of POMDP-based dialogue systems without state space compression. In *Proc. of the IEEE Workshop on Spoken Language Technology*, Miami, Florida, USA.

M. Gašić, C. Breslin, M. Henderson, D. Kim, M. Szummer, B. Thomson, P. Tsiakoulis, and S. Young. 2013. On-line policy optimisation of Bayesian spoken dialogue systems via human interaction. In *Proc. of the International Conference on Acoustics, Speech and Signal Processing*, Vancouver, Canada.

Kallirroi Georgila and David Traum. 2011. Reinforcement learning of argumentation dialogue policies in negotiation. In *Proc. of Interspeech*, Florence, Italy.

Kallirroi Georgila, James Henderson, and Oliver Lemon. 2006. User simulation for spoken dialogue systems: Learning and evaluation. In *Proc. of Interspeech*, Pittsburgh, Pennsylvania, USA.

Kallirroi Georgila, Maria K. Wolters, and Johanna D. Moore. 2010. Learning dialogue strategies from older and younger simulated users. In *Proc. of the Annual SIGdial Meeting on Discourse and Dialogue*, Tokyo, Japan.

Kallirroi Georgila. 2013. Reinforcement learning of two-issue negotiation dialogue policies. In *Proc. of the Annual SIGdial Meeting on Discourse and Dialogue*, Metz, France.

Peter A. Heeman. 2009. Representing the reinforcement learning state in a negotiation dialogue. In *Proc. of the IEEE Automatic Speech Recognition and Understanding Workshop*, Merano, Italy.

James Henderson, Oliver Lemon, and Kallirroi Georgila. 2008. Hybrid reinforcement/supervised learning of dialogue policies from fixed datasets. *Computational Linguistics*, 34(4):487–511.

Junling Hu and Michael P. Wellman. 1998. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proc. of the International Conference on Machine Learning*, Madison, Wisconsin, USA.

Filip Jurčíček, Blaise Thomson, and Steve Young. 2012. Reinforcement learning for parameter estimation in statistical spoken dialogue systems. *Computer Speech and Language*, 26(3):168–192.

Michail G. Lagoudakis and Ronald Parr. 2003. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149.

Oliver Lemon, Kallirroi Georgila, and James Henderson. 2006. Evaluating effectiveness and portability of reinforcement learned dialogue strategies with real users: The TALK TownInfo evaluation. In *Proc. of the IEEE Workshop on Spoken Language Technology*, Palm Beach, Aruba.

Lihong Li, Jason D. Williams, and Suhrid Balakrishnan. 2009. Reinforcement learning for dialog management using least-squares policy iteration and fast feature selection. In *Proc. of Interspeech*, Brighton, United Kingdom.

Michael L. Littman. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Proc. of the International Conference on Machine Learning*, New Brunswick, New Jersey, USA.

Yi Ma. 2013. User goal change model for spoken dialog state tracking. In *Proc. of the NAACL-HLT Student Research Workshop*, Atlanta, Georgia, USA.

Teruhisa Misu, Komei Sugiura, Kiyonori Ohtake, Chiori Hori, Hideki Kashioka, Hisashi Kawai, and Satoshi Nakamura. 2010. Modeling spoken decision making dialogue and optimization of its dialogue strategy. In *Proc. of the Annual SIGdial Meeting on Discourse and Dialogue*, Tokyo, Japan.

Teruhisa Misu, Kallirroi Georgila, Anton Leuski, and David Traum. 2012. Reinforcement learning of question-answering dialogue policies for virtual museum guides. In *Proc. of the Annual SIGdial Meeting on Discourse and Dialogue*, Seoul, South Korea.

Elnaz Nouri, Kallirroi Georgila, and David Traum. 2012. A cultural decision-making model for negotiation based on inverse reinforcement learning. In *Proc. of the Cognitive Science Conference*, Sapporo, Japan.

P. Paruchuri, N. Chakraborty, R. Zivan, K. Sycara, M. Dudik, and G. Gordon. 2009. POMDP based negotiation modeling. In *IJCAI Workshop on Modeling Intercultural Collaboration and Negotiation*, Pasadena, California, USA.

Olivier Pietquin and Helen Hastie. 2013. A survey on metrics for the evaluation of user simulations. *Knowledge Engineering Review*, 28(1):59–73.

Carl Edward Rasmussen and Christopher K. I. Williams. 2006. *Gaussian Processes for Machine Learning*. MIT Press.

Verena Rieser, Simon Keizer, Xingkun Liu, and Oliver Lemon. 2011. Adaptive information presentation for spoken dialogue systems: Evaluation with human subjects. In *Proc. of the European Workshop on Natural Language Generation*, Nancy, France.

Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. 2006. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *Knowledge Engineering Review*, 21(2):97–126.

Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. 2002. Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artificial Intelligence Research*, 16:105–133.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning: An Introduction*. MIT Press.

Joel R. Tetreault and Diane J. Litman. 2008. A reinforcement learning approach to evaluating state representations in spoken dialogue systems. *Speech Communication*, 50(8-9):683–696.

Blaise Thomson and Steve Young. 2010. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech and Language*, 24(4):562–588.

Jason D. Williams and Steve Young. 2007. Scaling POMDPs for spoken dialog management. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(7):2116–2129.

A Linear-Time Bottom-Up Discourse Parser with Constraints and Post-Editing

Vanessa Wei Feng
Department of Computer Science
University of Toronto
Toronto, ON, Canada
weifeng@cs.toronto.edu

Graeme Hirst
Department of Computer Science
University of Toronto
Toronto, ON, Canada
gh@cs.toronto.edu

Abstract

Text-level discourse parsing remains a challenge. The current state-of-the-art overall accuracy in relation assignment is 55.73%, achieved by Joty et al. (2013). However, their model has a high order of time complexity, and thus cannot be applied in practice. In this work, we develop a much faster model whose time complexity is linear in the number of sentences. Our model adopts a greedy bottom-up approach, with two linear-chain CRFs applied in cascade as local classifiers. To enhance the accuracy of the pipeline, we add additional constraints in the Viterbi decoding of the first CRF. In addition to efficiency, our parser also significantly outperforms the state of the art. Moreover, our novel approach of post-editing, which modifies a fully-built tree by considering information from constituents on upper levels, can further improve the accuracy.

1 Introduction

Discourse parsing is the task of identifying the presence and the type of the discourse relations between discourse units. While research in discourse parsing can be partitioned into several directions according to different theories and frameworks, Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) is probably the most ambitious one, because it aims to identify not only the discourse relations in a small local context, but also the hierarchical tree structure for the **full text**: from the relations relating the smallest discourse units (called elementary discourse units, EDUs), to the ones connecting paragraphs.

For example, Figure 1 shows a text fragment consisting of two sentences with four EDUs in total (e_1-e_4). Its discourse tree representation is shown below the text, following the notation convention of RST: the two EDUs e_1 and e_2 are related by a mononuclear relation CONSEQUENCE, where e_2 is the more salient span (called *nucleus*, and e_1 is called *satellite*); e_3 and e_4 are related by another mononuclear relation CIRCUMSTANCE, with e_4 as the nucleus; the two spans $e_{1:2}$ and $e_{3:4}$ are further related by a multi-nuclear relation SEQUENCE, with both spans as the nucleus.

Conventionally, there are two major sub-tasks related to text-level discourse parsing: (1) **EDU segmentation**: to segment the raw text into EDUs, and (2) **tree-building**: to build a discourse tree from EDUs, representing the discourse relations in the text. Since the first sub-task is considered relatively easy, with the state-of-art accuracy at above 90% (Joty et al., 2012), the recent research focus is on the second sub-task, and often uses manual EDU segmentation.

The current state-of-the-art overall accuracy of the tree-building sub-task, evaluated on the RST Discourse Treebank (RST-DT, to be introduced in Section 8), is 55.73% by Joty et al. (2013). However, as an optimal discourse parser, Joty et al.'s model is highly inefficient in practice, with respect to both their DCRF-based local classifiers, and their CKY-like bottom-up parsing algorithm. DCRF (Dynamic Conditional Random Fields) is a generalization of linear-chain CRFs, in which each time slice contains a set of state variables and edges (Sutton et al., 2007). CKY parsing is a bottom-up parsing algorithm which searches all possible parsing paths by dynamic programming. Therefore, despite its superior performance, their model is infeasible in most realistic situations.

The main objective of this work is to develop a more efficient discourse parser, with similar or even better performance with respect to Joty et al.'s optimal parser, but able to produce parsing results in real time.

Our contribution is three-fold. First, with a

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 511–521,
Baltimore, Maryland, USA, June 23-25 2014.

[**On Aug. 1, the state tore up its controls**,]e_1 [**and food prices leaped**]e_2 [**Without buffer stocks**,]e_3 [**inflation exploded**.]e_4

wsj_1146

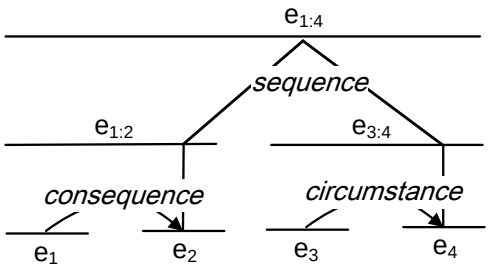


Figure 1: An example text fragment composed of two sentences and four EDUs, with its RST discourse tree representation shown below.

greedy bottom-up strategy, we develop a discourse parser with a time complexity linear in the total number of sentences in the document. As a result of successfully avoiding the expensive non-greedy parsing algorithms, our discourse parser is very efficient in practice. Second, by using two linear-chain CRFs to label a sequence of discourse constituents, we can incorporate contextual information in a more natural way, compared to using traditional discriminative classifiers, such as SVMs. Specifically, in the Viterbi decoding of the first CRF, we include additional constraints elicited from common sense, to make more effective local decisions. Third, after a discourse (sub)tree is fully built from bottom up, we perform a novel post-editing process by considering information from the constituents on upper levels. We show that this post-editing can further improve the overall parsing performance.

2 Related work

2.1 HILDA discourse parser

The HILDA discourse parser by Hernault et al. (2010) is the first attempt at RST-style text-level discourse parsing. It adopts a pipeline framework, and greedily builds the discourse tree from the bottom up. In particular, starting from EDUs, at each step of the tree-building, a binary SVM classifier is first applied to determine which pair of adjacent discourse constituents should be merged to form a larger span, and another multi-class SVM classifier is then applied to assign the type of discourse relation that holds between the chosen pair.

The strength of HILDA's greedy tree-building strategy is its efficiency in practice. Also, the employment of SVM classifiers allows the incorporation of rich features for better data representation (Feng and Hirst, 2012). However, HILDA's approach also has obvious weakness: the greedy algorithm may lead to poor performance due to local optima, and more importantly, the SVM classifiers are not well-suited for solving structural problems due to the difficulty of taking context into account.

2.2 Joty et al.'s joint model

Joty et al. (2013) approach the problem of text-level discourse parsing using a model trained by Conditional Random Fields (CRF). Their model has two distinct features.

First, they decomposed the problem of text-level discourse parsing into two stages: intra-sentential parsing to produce a discourse tree for each sentence, followed by multi-sentential parsing to combine the sentence-level discourse trees and produce the text-level discourse tree. Specifically, they employed two separate models for intra- and multi-sentential parsing. Their choice of two-stage parsing is well motivated for two reasons: (1) it has been shown that sentence boundaries correlate very well with discourse boundaries, and (2) the scalability issue of their CRF-based models can be overcome by this decomposition.

Second, they jointly modeled the structure and the relation for a given pair of discourse units. For example, Figure 2 shows their intra-sentential model, in which they use the bottom layer to represent discourse units; the middle layer of binary nodes to predict the connection of adjacent discourse units; and the top layer of multi-class nodes to predict the type of the relation between two units. Their model assigns a probability to each possible constituent, and a CKY-like parsing algorithm finds the globally optimal discourse tree, given the computed probabilities.

The strength of Joty et al.'s model is their joint modeling of the structure and the relation, such that information from each aspect can interact with the other. However, their model has a major defect in its inefficiency, or even infeasibility, for application in practice. The inefficiency lies in both their DCRF-based joint model, on which inference is usually slow, and their CKY-like parsing algorithm, whose issue is more prominent. Due to the $O(n^3)$ time complexity, where n is the number

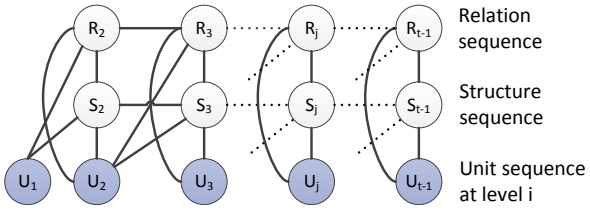


Figure 2: Joty et al. (2013)'s intra-sentential Condition Random Fields.

of input discourse units, for large documents, the parsing simply takes too long[1].

3 Overall work flow

Figure 3 demonstrates the overall work flow of our discourse parser. The general idea is that, similar to Joty et al. (2013), we perform a sentence-level parsing for each sentence first, followed by a text-level parsing to generate a full discourse tree for the whole document. However, in addition to efficiency (to be shown in Section 6), our discourse parser has a distinct feature, which is the post-editing component (to be introduced in Section 5), as outlined in dashes.

Our discourse parser works as follows. A document D is first segmented into a list of sentences. Each sentence S_i, after being segmented into EDUs (not shown in the figure), goes through an intra-sentential bottom-up tree-building model M_{intra}, to form a sentence-level discourse tree T_{S_i}, with the EDUs as leaf nodes. After that, we apply the intra-sentential post-editing model P_{intra} to modify the generated tree T_{S_i} to $T^p_{S_i}$, by considering upper-level information.

We then combine all sentence-level discourse tree $T^p_{S_i}$'s using our multi-sentential bottom-up tree-building model M_{multi} to generate the text-level discourse tree T_D. Similar to sentence-level parsing, we also post-edit T_D using P_{multi} to produce the final discourse tree T^p_D.

[1]The largest document in the RST-DT contains over 180 sentences, i.e., $n > 180$ for their multi-sentential CKY parsing. Intuitively, suppose the average time to compute the probability of each constituent is 0.01 second, then in total, the CKY-like parsing takes over 16 hours. It is possible to optimize Joty et al.'s CKY-like parsing by replacing their CRF-based computation for upper-level constituents with some local computation based on the probabilities of lower-level constituents. However, such optimization is beyond the scope of this paper.

4 Bottom-up tree-building

For both intra- and multi-sentential parsing, our bottom-up tree-building process adopts a similar greedy pipeline framework like the HILDA discourse parser (discussed in Section 2.1), to guarantee efficiency for large documents. In particular, starting from the constituents on the bottom level (EDUs for intra-sentential parsing and sentence-level discourse trees for multi-sentential parsing), at each step of the tree-building, we greedily merge a pair of adjacent discourse constituents such that the merged constituent has the highest probability as predicted by our *structure* model. The *relation* model is then applied to assign the relation to the new constituent.

4.1 Linear-chain CRFs as Local models

Now we describe the local models we use to make decisions for a given pair of adjacent discourse constituents in the bottom-up tree-building. There are two dimensions for our local models: (1) scope of the model: intra- or multi-sentential, and (2) purpose of the model: for determining structures or relations. So we have four local models, M^{struct}_{intra}, M^{rel}_{intra}, M^{struct}_{multi}, and M^{rel}_{multi}.

While our bottom-up tree-building shares the greedy framework with HILDA, unlike HILDA, our local models are implemented using CRFs. In this way, we are able to take into account the sequential information from contextual discourse constituents, which cannot be naturally represented in HILDA with SVMs as local classifiers. Therefore, our model incorporates the strengths of both HILDA and Joty et al.'s model, i.e., the efficiency of a greedy parsing algorithm, and the ability to incorporate sequential information with CRFs.

As shown by Feng and Hirst (2012), for a pair of discourse constituents of interest, the sequential information from contextual constituents is crucial for determining structures. Therefore, it is well motivated to use Conditional Random Fields (CRFs) (Lafferty et al., 2001), which is a discriminative probabilistic graphical model, to make predictions for a sequence of constituents surrounding the pair of interest.

In this sense, our local models appear similar to Joty et al.'s non-greedy parsing models. However, the major distinction between our models and theirs is that we do not jointly model the structure and the relation; rather, we use two linear-

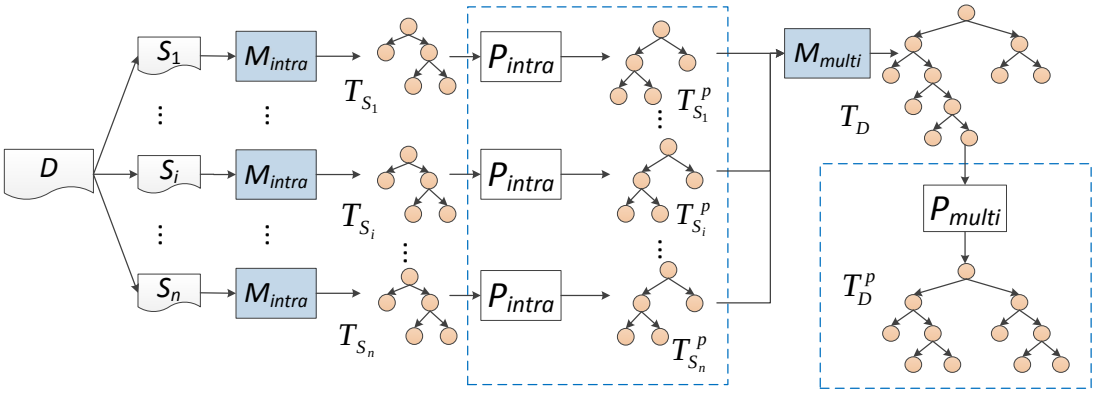


Figure 3: The work flow of our proposed discourse parser. In the figure, M_{intra} and M_{multi} stand for the intra- and multi-sentential bottom-up tree-building models, and P_{intra} and P_{multi} stand for the intra- and multi-sentential post-editing models.

chain CRFs to model the structure and the relation separately. Although joint modeling has shown to be effective in various NLP and computer vision applications (Sutton et al., 2007; Yang et al., 2009; Wojek and Schiele, 2008), our choice of using two separate models is for the following reasons:

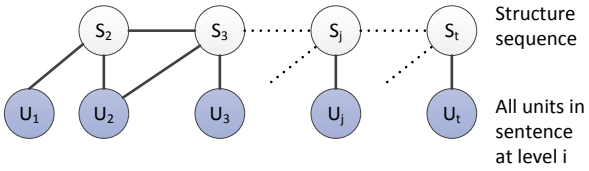
First, it is not entirely appropriate to model the structure and the relation at the same time. For example, with respect to Figure 2, it is unclear how the relation node R_j is represented for a training instance whose structure node $S_j = 0$, i.e., the units U_{j-1} and U_j are disjoint. Assume a special relation NO-REL is assigned for R_j. Then, in the tree-building process, we will have to deal with the situations where the joint model yields conflicting predictions: it is possible that the model predicts $S_j = 1$ and $R_j =$ NO-REL, or vice versa, and we will have to decide which node to trust (and thus in some sense, the structure and the relation is no longer jointly modeled).

Secondly, as a joint model, it is mandatory to use a dynamic CRF, for which exact inference is usually intractable or slow. In contrast, for linear-chain CRFs, efficient algorithms and implementations for exact inference exist.

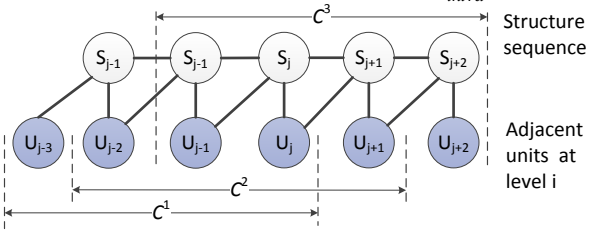
4.2 Structure models

4.2.1 Intra-sentential structure model

Figure 4a shows our intra-sentential structure model M^{struct}_{intra} in the form of a linear-chain CRF. Similar to Joty et al.'s intra-sentential model, the first layer of the chain is composed of discourse constituents U_j's, and the second layer is composed of binary nodes S_j's to indicate the probability of merging adjacent discourse constituents.



(a) Intra-sentential structure model M^{struct}_{intra}.



(b) Multi-sentential structure model M^{struct}_{multi}. C^1, C^2, and C^3 denote the three chains for predicting U_j and U_{j+1}.

Figure 4: Local structure models.

At each step in the bottom-up tree-building process, we generate a single sequence E, consisting of $U_1, U_2, \ldots, U_j, \ldots, U_t$, which are all the current discourse constituents in the sentence that need to be processed. For instance, initially, we have the sequence $E_1 = \{e_1, e_2, \ldots, e_m\}$, which are the EDUs of the sentence; after merging e_1 and e_2 on the second level, we have $E_2 = \{e_{1:2}, e_3, \ldots, e_m\}$; after merging e_4 and e_5 on the third level, we have $E_3 = \{e_{1:2}, e_3, e_{4:5}, \ldots, e_m\}$, and so on.

Because the structure model is the first component in our pipeline of local models, its accuracy is crucial. Therefore, to improve its accuracy, we enforce additional commonsense constraints in its Viterbi decoding. In particular, we disallow 1-1 transitions between adjacent labels (a discourse unit can be merged with at most one adjacent unit), and we disallow all-zero sequences (at least one

pair must be merged).

Since the computation of E_i **does not** depend on a particular pair of constituents, we can use the same sequence E_i to compute structural probabilities for **all** adjacent constituents. In contrast, Joty et al.'s computation of intra-sentential sequences depends on the particular pair of constituents: the sequence is composed of the pair in question, with other EDUs in the sentence, even if those EDUs have already been merged. Thus, different CRF chains have to be formed for different pairs of constituents. In addition to efficiency, our use of a single CRF chain for all constituents can better capture the sequential dependencies among context, by taking into account the information from partially built discourse constituents, rather than bottom-level EDUs only.

4.2.2 Multi-sentential structure model

For multi-sentential parsing, where the smallest discourse units are single sentences, as argued by Joty et al. (2013), it is not feasible to use a long chain to represent all constituents, due to the fact that it takes $O(TM^2)$ time to perform the forward-backward exact inference on a chain with T units and an output vocabulary size of M, thus the overall complexity for all possible sequences in their model is $O(M^2n^3)$[2].

Instead, we choose to take a sliding-window approach to form CRF chains for a particular pair of constituents, as shown in Figure 4b. For example, suppose we wish to compute the structural probability for the pair U_{j-1} and U_j, we form three chains, each of which contains two contextual constituents: $C^1 = \{U_{j-3}, U_{j-2}, U_{j-1}, U_j\}$, $C^2 = \{U_{j-2}, U_{j-1}, U_j, U_{j+1}\}$, and $C^3 = \{U_{j-1}, U_j, U_{j+1}, U_{j+2}\}$. We then find the chain $C^t, 1 \leq t \leq 3$, with the highest joint probability over the entire sequence, and assign its marginal probability $P(S_j^t = 1)$ to $P(S_j = 1)$.

Similar to M_{intra}^{struct}, for M_{multi}^{struct}, we also include additional constraints in the Viterbi decoding, by disallowing transitions between two ones, and disallowing the sequence to be all zeros if it contains all the remaining constituents in the document.

4.3 Relation models

4.3.1 Intra-sentential relation model

The intra-sentential relation model M_{intra}^{rel}, shown in Figure 5a, works in a similar way to M_{intra}^{struct}, as described in Section 4.2.1. The linear-chain CRF contains a first layer of all discourse constituents U_j's in the sentence on level i, and a second layer of relation nodes R_j's to represent the relation between a pair of discourse constituents.

However, unlike the structure model, adjacent relation nodes do not share discourse constituents on the first layer. Rather, each relation node R_j attempts to model the relation of one single constituent U_j, by taking U_j's left and right subtrees $U_{j,L}$ and $U_{j,R}$ as its first-layer nodes; if U_j is a single EDU, then the first-layer node of R_j is simply U_j, and R_j is a special relation symbol LEAF[3]. Since we know, a priori, that the constituents in the chains are either leaf nodes or the ones that have been merged by our structure model, we never need to worry about the NO-REL issue as outlined in Section 4.1.

In the bottom-up tree-building process, after merging a pair of adjacent constituents using M_{intra}^{struct} into a new constituent, say U_j, we form a chain consisting of all current constituents in the sentence to decide the relation label for U_j, i.e., the R_j node in the chain. In fact, by performing inference on this chain, we produce predictions not only for R_j, but also for all other R nodes in the chain, which correspond to all other constituents in the sentence. Since those non-leaf constituents are already labeled in previous steps in the tree-building, we can now **re-assign** their relations if the model predicts differently in this step. Therefore, this re-labeling procedure can compensate for the loss of accuracy caused by our greedy bottom-up strategy to some extent.

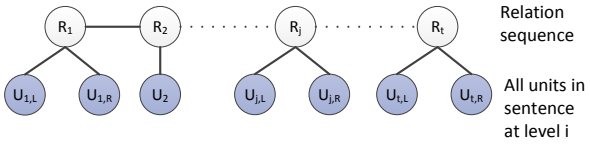
4.3.2 Multi-sentential relation model

Figure 5b shows our multi-sentential relation model. Like M_{intra}^{rel}, the first layer consists of adjacent discourse units, and the relation nodes on the second layer model the relation of each constituent separately.

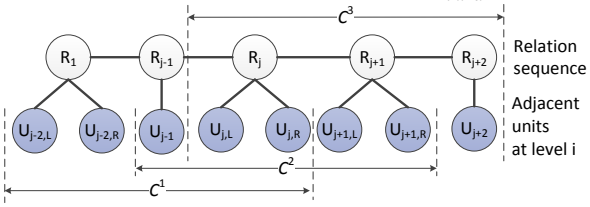
Similar to M_{multi}^{struct} introduced in Section 4.2.2, M_{multi}^{rel} also takes a sliding-window approach to predict labels for constituents in a local context. For a constituent U_j to be predicted, we form three chains, and use the chain with the highest joint probability to assign or re-assign relations to constituents in that chain.

[2]The time complexity will be reduced to $O(M^2n^2)$, if we use the same chain for all constituents as in our M_{intra}^{struct}.

[3]These leaf constituents are represented using a special feature vector `is_leaf = True`; thus the CRF never labels them with relations other than LEAF.



(a) Intra-sentential relation model M^{rel}_{intra}.



(b) Multi-sentential relation model M^{rel}_{multi}. C^1, C^2, and C^3 denote the three sliding windows for predicting $U_{j,L}$ and $U_{j,R}$.

Figure 5: Local relation models.

5 Post-editing

After an intra- or multi-sentential discourse tree is fully built, we perform a post-editing to consider possible modifications to the current tree, by considering useful information from the discourse constituents on upper levels, which is unavailable in the bottom-up tree-building process.

The motivation for post-editing is that, some particular discourse relations, such as TEXTUAL-ORGANIZATION, tend to occur on the top levels of the discourse tree; thus, information such as the depth of the discourse constituent can be quite indicative. However, the exact depth of a discourse constituent is usually unknown in the bottom-up tree-building process; therefore, it might be beneficial to modify the tree by including top-down information after the tree is fully built.

The process of post-editing is shown in Algorithm 1. For each input discourse tree T, which is already fully built by bottom-up tree-building models, we do the following:

Lines 3 – 9: Identify the lowest level of T on which the constituents can be modified according to the post-editing structure component, P^{struct}. To do so, we maintain a list L to store the discourse constituents that need to be examined. Initially, L consists of all the bottom-level constituents in T. At each step of the loop, we consider merging the pair of adjacent units in L with the highest probability predicted by P^{struct}. If the predicted pair is not merged in the original tree T, then a possible modification is located; otherwise, we merge the pair, and proceed to the next iteration.

Lines 10 – 12: If modifications have been proposed in the previous step, we build a new tree

Algorithm 1 Post-editing algorithm.

Input: A fully built discourse tree T.

1: **if** $|T| = 1$ **then**
2: **return** T ▷ Do nothing if it is a single EDU.
3: $L \leftarrow [U_1, U_2, \ldots, U_t]$ ▷ The bottom-level constituents in T.
4: **while** $|L| > 2$ **do**
5: $i \leftarrow \text{PREDICTMERGING}(L, P^{struct})$
6: $p \leftarrow \text{PARENT}(L[i], L[i+1], T)$
7: **if** p = NULL **then**
8: **break**
9: Replace $L[i]$ and $L[i+1]$ with p
10: **if** $|L| = 2$ **then**
11: $L \leftarrow [U_1, U_2, \ldots, U_t]$
12: $T^p \leftarrow \text{BUILDTREE}(L, P^{struct}, P^{rel}, T)$

Output: T^p

T^p using P^{struct} as the structure model, and P^{rel} as the relation model, from the constituents on which modifications are proposed. Otherwise, T^p is built from the bottom-level constituents of T. The upper-level information, such as the depth of a discourse constituent, is derived from the initial tree T.

5.1 Local models

The local models, $P^{\{struct|rel\}}_{\{intra|multi\}}$, for post-editing is almost identical to their counterparts of the bottom-up tree-building, except that the linear-chain CRFs in post-editing includes additional features to represent information from constituents on higher levels (to be introduced in Section 7).

6 Linear time complexity

Here we analyze the time complexity of each component in our discourse parser, to quantitatively demonstrate the time efficiency of our model. The following analysis is focused on the bottom-up tree-building process, but a similar analysis can be carried out for the post-editing process. Since the number of operations in the post-editing process is roughly the same (1.5 times in the worst case) as in the bottom-up tree-building, post-editing shares the same complexity as the tree-building.

6.1 Intra-sentential parsing

Suppose the input document is segmented into n sentences, and each sentence S_k contains m_k EDUs. For each sentence S_k with m_k EDUs, the

overall time complexity to perform intra-sentential parsing is $O(m_k^2)$. The reason is the following. On level i of the bottom-up tree-building, we generate a single chain to represent the structure or relation for all the $m_k - i$ constituents that are currently in the sentence. The time complexity for performing forward-backward inference on the single chain is $O((m_k - i) \times M^2) = O(m_k - i)$, where the constant M is the size of the output vocabulary. Starting from the EDUs on the bottom level, we need to perform inference for one chain on each level during the bottom-up tree-building, and thus the total time complexity is $\Sigma_{i=1}^{m_k} O(m_k - i) = O(m_k^2)$.

The total time to generate sentence-level discourse trees for n sentences is $\Sigma_{k=1}^{n} O(m_k^2)$. It is fairly safe to assume that each m_k is a constant, in the sense that m_k is independent of the total number of sentences in the document. Therefore, the total time complexity $\Sigma_{k=1}^{n} O(m_k^2) \leq n \times O(\max_{1 \leq j \leq n}(m_j^2)) = n \times O(1) = O(n)$, i.e., linear in the total number of sentences.

6.2 Multi-sentential parsing

For multi-sentential models, M_{multi}^{struct} and M_{multi}^{rel}, as shown in Figures 4b and 5b, for a pair of constituents of interest, we generate multiple chains to predict the structure or the relation.

By including a constant number k of discourse units in each chain, and considering a constant number l of such chains for computing each adjacent pair of discourse constituents ($k = 4$ for M_{multi}^{struct} and $k = 3$ for M_{multi}^{rel}; $l = 3$), we have an overall time complexity of $O(n)$. The reason is that it takes $l \times O(kM^2) = O(1)$ time, where l, k, M are all constants, to perform exact inference for a given pair of adjacent constituents, and we need to perform such computation for all $n - 1$ pairs of adjacent sentences on the first level of the tree-building. Adopting a greedy approach, on an arbitrary level during the tree-building, once we decide to merge a certain pair of constituents, say U_j and U_{j+1}, we only need to recompute a small number of chains, i.e., the chains which originally include U_j or U_{j+1}, and inference on each chain takes $O(1)$. Therefore, the total time complexity is $(n-1) \times O(1) + (n-1) \times O(1) = O(n)$, where the first term in the summation is the complexity of computing all chains on the bottom level, and the second term is the complexity of computing the constant number of chains on higher levels.

We have thus showed that the time complexity is *linear* in n, which is the number of sentences in the document. In fact, under the assumption that the number of EDUs in each sentence is independent of n, it can be shown that the time complexity is also linear in the total number of EDUs[4].

7 Features

In our local models, to encode two adjacent units, U_j and U_{j+1}, within a CRF chain, we use the following 10 sets of features, some of which are modified from Joty et al.'s model.

Organization features: Whether U_j (or U_{j+1}) is the first (or last) constituent in the sentence (for intra-sentential models) or in the document (for multi-sentential models); whether U_j (or U_{j+1}) is a bottom-level constituent.

Textual structure features: Whether U_j contains more sentences (or paragraphs) than U_{j+1}.

N-gram features: The beginning (or end) lexical n-grams in each unit; the beginning (or end) POS n-grams in each unit, where $n \in \{1, 2, 3\}$.

Dominance features: The PoS tags of the head node and the attachment node; the lexical heads of the head node and the attachment node; the dominance relationship between the two units.

Contextual features: The feature vector of the previous and the next constituent in the chain.

Substructure features: The root node of the left and right discourse subtrees of each unit.

Syntactic features: whether each unit corresponds to a single syntactic subtree, and if so, the top PoS tag of the subtree; the distance of each unit to their lowest common ancestor in the syntax tree (intra-sentential only).

Entity transition features: The type and the number of entity transitions across the two units. We adopt Barzilay and Lapata (2008)'s entity-based local coherence model to represent a document by an entity grid, and extract local transitions among entities in continuous discourse constituents. We use bigram and trigram transitions with syntactic roles attached to each entity.

[4] We implicitly made an assumption that the parsing time is dominated by the time to perform inference on CRF chains. However, for complex features, the time required for feature computation might be dominant. Nevertheless, a careful caching strategy can accelerate feature computation, since a large number of multi-sentential chains overlap with each other.

Cue phrase features: Whether a cue phrase occurs in the first or last EDU of each unit. The cue phrase list is based on the connectives collected by Knott and Dale (1994)

Post-editing features: The depth of each unit in the initial tree.

8 Experiments

For pre-processing, we use the Stanford CoreNLP (Klein and Manning, 2003; de Marneffe et al., 2006; Recasens et al., 2013) to syntactically parse the texts and extract coreference relations, and we use Penn2Malt[5] to lexicalize syntactic trees to extract dominance features.

For local models, our structure models are trained using MALLET (McCallum, 2002) to include constraints over transitions between adjacent labels, and our relation models are trained using CRFSuite (Okazaki, 2007), which is a fast implementation of linear-chain CRFs.

The data that we use to develop and evaluate our discourse parser is the RST Discourse Treebank (RST-DT) (Carlson et al., 2001), which is a large corpus annotated in the framework of RST. The RST-DT consists of 385 documents (347 for training and 38 for testing) from the *Wall Street Journal*. Following previous work on the RST-DT (Hernault et al., 2010; Feng and Hirst, 2012; Joty et al., 2012; Joty et al., 2013), we use 18 coarse-grained relation classes, and with nuclearity attached, we have a total set of 41 distinct relations. Non-binary relations are converted into a cascade of right-branching binary relations.

9 Results and Discussion

9.1 Parsing accuracy

We compare four different models using manual EDU segmentation. In Table 1, the *j*CRF model in the first row is the optimal CRF model proposed by Joty et al. (2013). gSVM^{FH} in the second row is our implementation of HILDA's greedy parsing algorithm using Feng and Hirst (2012)'s enhanced feature set. The third model, *g*CRF, represents our greedy CRF-based discourse parser, and the last row, gCRF^{PE}, represents our parser with the post-editing component included.

In order to conduct a direct comparison with Joty et al.'s model, we use the same set of evaluation metrics, i.e., the unlabeled and labeled precision, recall, and F-score[6] as defined by Marcu (2000). For evaluating relations, since there is a skewed distribution of different relation types in the corpus, we also include the macro-averaged F1-score (MAFS)[7] as another metric, to emphasize the performance of infrequent relation types. We report the MAFS separately for the correctly retrieved constituents (i.e., the span boundary is correct) and all constituents in the reference tree.

Model	Span	Nuc	Relation	
			Acc	MAFS
*j*CRF	82.5	68.4	55.7	N/A
gSVM^{FH}	82.8	67.1	52.0	27.4/23.3
*g*CRF	84.9*	69.9*	57.2*	35.3/31.3
gCRF^{PE}	**85.7***†	**71.0***†	**58.2***†	**36.2/32.3**
Human	88.7	77.7	65.8	N/A

$*$: significantly better than gSVM^{FH} ($p < .01$)
$\dagger$: significantly better than *g*CRF ($p < .01$)

Table 1: Performance of different models using gold-standard EDU segmentation, evaluated using the constituent accuracy (%) for span, nuclearity, and relation. For relation, we also report the macro-averaged F1-score (MAFS) for correctly retrieved constituents (before the slash) and for all constituents (after the slash). Statistical significance is verified using Wilcoxon's signed-rank test.

As demonstrated by Table 1, our greedy CRF models perform significantly better than the other two models. Since we do not have the actual output of Joty et al.'s model, we are unable to conduct significance testing between our models and theirs. But in terms of overall accuracy, our *g*CRF model outperforms their model by 1.5%. Moreover, with post-editing enabled, gCRF^{PE} significantly ($p < .01$) outperforms our initial model *g*CRF by another 1% in relation assignment, and this overall accuracy of 58.2% is close to 90% of human performance. With respect to the macro-averaged F1-scores, adding the post-editing component also obtains about 1% improvement.

However, the overall MAFS is still at the lower

[5] http://stp.lingfil.uu.se/~nivre/research/Penn2Malt.html.

[6] For manual segmentation, precision, recall, and F-score are the same.

[7] MAFS is the F1-score averaged among all relation classes by equally weighting each class. Therefore, we cannot conduct significance test between different MAFS.

	Avg	Min	Max
# of EDUs	61.74	4	304
# of Sentences	26.11	2	187
# of EDUs per sentence	2.36	1	10

Table 2: Characteristics of the 38 documents in the test data.

end of 30% for all constituents. Our error analysis shows that, for two relation classes, TOPIC-CHANGE and TEXTUAL-ORGANIZATION, our model fails to retrieve any instance, and for TOPIC-COMMENT and EVALUATION, our model scores a class-wise F_1 score lower than 5%. These four relation classes, apart from their infrequency in the corpus, are more abstractly defined, and thus are particularly challenging.

9.2 Parsing efficiency

We further illustrate the efficiency of our parser by demonstrating the time consumption of different models.

First, as shown in Table 2, the average number of sentences in a document is 26.11, which is already too large for optimal parsing models, e.g., the CKY-like parsing algorithm in *j*CRF, let alone the fact that the largest document contains several hundred of EDUs and sentences. Therefore, it should be seen that non-optimal models are required in most cases.

In Table 3, we report the parsing time[8] for the last three models, since we do not know the time of *j*CRF. Note that the parsing time excludes the time cost for any necessary pre-processing. As can be seen, our *g*CRF model is considerably faster than *g*SVM^{FH}, because, on one hand, feature computation is expensive in *g*SVM^{FH}, since *g*SVM^{FH} utilizes a rich set of features; on the other hand, in *g*CRF, we are able to accelerate decoding by multi-threading MALLET (we use four threads). Even for the largest document with 187 sentences, *g*CRF is able to produce the final tree after about 40 seconds, while *j*CRF would take over 16 hours assuming each DCRF decoding takes only 0.01 second. Although enabling post-editing doubles the time consumption, the overall time is still acceptable in practice, and the loss of efficiency can be compensated by the improvement in accuracy.

[8]Tested on a Linux system with four duo-core 3.0GHz processors and 16G memory.

Model	**Parsing Time (seconds)**		
	Avg	Min	Max
*g*SVM^{FH}	11.19	0.42	124.86
*g*CRF	5.52	0.05	40.57
*g*CRF^{PE}	10.71	0.12	84.72

Table 3: The parsing time (in seconds) for the 38 documents in the test set of RST-DT. Time cost of any pre-processing is excluded from the analysis.

10 Conclusions

In this paper, we presented an efficient text-level discourse parser with time complexity linear in the total number of sentences in the document. Our approach was to adopt a greedy bottom-up tree-building, with two linear-chain CRFs as local probabilistic models, and enforce reasonable constraints in the first CRF's Viterbi decoding. While significantly outperforming the state-of-the-art model by Joty et al. (2013), our parser is much faster in practice. In addition, we propose a novel idea of post-editing, which modifies a fully-built discourse tree by considering information from upper-level constituents. We show that, although doubling the time consumption, post-editing can further boost the parsing performance to close to 90% of human performance.

In future work, we wish to further explore the idea of post-editing, since currently we use only the depth of the subtrees as upper-level information. Moreover, we wish to study whether we can incorporate constraints into the relation models, as we do to the structure models. For example, it might be helpful to train the relation models using additional criteria, such as Generalized Expectation (Mann and McCallum, 2008), to better take into account some prior knowledge about the relations. Last but not least, as reflected by the low MAFS in our experiments, some particularly difficult relation types might need specifically designed features for better recognition.

Acknowledgments

We thank Professor Gerald Penn and the reviewers for their valuable advice and comments. This work was financially supported by the Natural Sciences and Engineering Research Council of Canada and by the University of Toronto.

References

Jason Baldridge and Alex Lascarides. 2005. Probabilistic head-driven parsing for discourse structure. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 96–103, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: an entity-based approach. *Computational Linguistics*, 34(1):1–34.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2001. Building a discourse-tagged corpus in the framework of Rhetorical Structure Theory. In *Proceedings of Second SIGDial Workshop on Discourse and Dialogue (SIGDial 2001)*, pages 1–10.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*.

Vanessa Wei Feng and Graeme Hirst. 2012. Text-level discourse parsing with rich linguistic features. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL 2012)*, pages 60–68, Jeju, Korea.

Hugo Hernault, Helmut Prendinger, David A. duVerle, and Mitsuru Ishizuka. 2010. HILDA: A discourse parser using support vector machine classification. *Dialogue and Discourse*, 1(3):1–33.

Shafiq Joty, Giuseppe Carenini, and Raymond T. Ng. 2012. A novel discriminative framework for sentence-level discourse analysis. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL 2012, pages 904–915.

Shafiq Joty, Giuseppe Carenini, Raymond Ng, and Yashar Mehdad. 2013. Combining intra- and multi-sentential rhetorical parsing for document-level discourse analysis. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pages 486–496, Sofia, Bulgaria, August.

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

Alistair Knott and Robert Dale. 1994. Using linguistic phenomena to motivate a set of coherence relations. *Discourse Processes*, 18(1):35–64.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML 2001, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Gideon S. Mann and Andrew McCallum. 2008. Generalized Expectation Criteria for semi-supervised learning of Conditional Random Fields. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL 2008)*, pages 870–878, Columbus, Ohio, June. Association for Computational Linguistics.

William Mann and Sandra Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Daniel Marcu. 2000. *The Theory and Practice of Discourse Parsing and Summarization*. The MIT Press.

Andrew Kachites McCallum. 2002. MALLET: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Philippe Muller, Stergos Afantenos, Pascal Denis, and Nicholas Asher. 2012. Constrained decoding for text-level discourse parsing. In *Proceedings of COLING 2012*, pages 1883–1900, Mumbai, India, December. The COLING 2012 Organizing Committee.

Naoaki Okazaki. 2007. CRFsuite: a fast implementation of conditional random fields (CRFs). http://www.chokkan.org/software/crfsuite/.

Marta Recasens, Marie-Catherine de Marneffe, and Christopher Potts. 2013. The life and death of discourse entities: Identifying singleton mentions. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 627–633, Atlanta, Georgia, June. Association for Computational Linguistics.

Rajen Subba and Barbara Di Eugenio. 2009. An effective discourse parser that uses rich linguistic information. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 566–574, Boulder, Colorado, June. Association for Computational Linguistics.

Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. 2007. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *The Journal of Machine Learning Research*, 8:693–723, May.

Christian Wojek and Bernt Schiele. 2008. A dynamic conditional random field model for joint labeling of object and scene classes. In *European Conference*

on Computer Vision (ECCV 2008), pages 733–747, Marseille, France.

Dong Yang, Paul Dixon, Yi-Cheng Pan, Tasuku Oonishi, Masanobu Nakamura, and Sadaoki Furui. 2009. Combining a two-step conditional random field model and a joint source channel model for machine transliteration. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 72–75, Suntec, Singapore, August. Association for Computational Linguistics.

Negation Focus Identification with Contextual Discourse Information

Bowei Zou **Qiaoming Zhu** **Guodong Zhou***
Natural Language Processing Lab, School of Computer Science and Technology
Soochow University, Suzhou, 215006, China
zoubowei@gmail.com, {qmzhu, gdzhou}@suda.edu.cn

Abstract

Negative expressions are common in natural language text and play a critical role in information extraction. However, the performances of current systems are far from satisfaction, largely due to its focus on intra-sentence information and its failure to consider inter-sentence information. In this paper, we propose a graph model to enrich intra-sentence features with inter-sentence features from both lexical and topic perspectives. Evaluation on the *SEM 2012 shared task corpus indicates the usefulness of contextual discourse information in negation focus identification and justifies the effectiveness of our graph model in capturing such global information.

1 Introduction

Negation is a grammatical category which comprises various kinds of devices to reverse the truth value of a proposition (Morante and Sporleder, 2012). For example, sentence (1) could be interpreted as *it is not the case that he stopped*.

(1) *He didn't stop.*

Negation expressions are common in natural language text. According to the statistics on biomedical literature genre (Vincze et al., 2008), 19.44% of sentences contain negative expressions. The percentage rises to 22.5% on Conan Doyle stories (Morante and Daelemans, 2012). It is interesting that a negative sentence may have both negative and positive meanings. For example, sentence (2) could be interpreted as *He stopped, but not until he got to Jackson Hole* with positive part *he stopped* and negative part *until he got to Jackson Hole*. Moreover, a negative expression normally interacts with some special part in the sentence, referred as negation focus in linguistics. Formally, negation focus is defined as the special part in the sentence, which is most prominently or explicitly negated by a negative expression. Hereafter, we denote negative expression in ***boldface*** and negation focus *underlined*.

(2) *He did**n't** stop until he got to Jackson Hole.*

While people tend to employ stress or intonation in speech to emphasize negation focus and thus it is easy to identify negation focus in speech corpora, such stress or intonation information often misses in the dominating text corpora. This poses serious challenges on negation focus identification. Current studies (e.g., Blanco and Moldovan, 2011; Rosenberg and Bergler, 2012) sort to various kinds of intra-sentence information, such as lexical features, syntactic features, semantic role features and so on, ignoring less-obvious inter-sentence information. This largely defers the performance of negation focus identification and its wide applications, since such contextual discourse information plays a critical role on negation focus identification. Take following sentence as an example.

(3) *Helen did**n't** allow her youngest son to play the violin.*

In sentence (3), there are several scenarios on identification of negation focus, with regard to negation expression ***n't***, given different contexts:
Scenario A: Given sentence *But her husband did* as next sentence, the negation focus should be *Helen*, yielding interpretation *the person who didn't allow the youngest son to play the violin is Helen but not her husband*.
Scenario B: Given sentence *She thought that he didn't have the artistic talent like her eldest son* as next sentence, the negation focus should be *the youngest son*, yielding interpretation *Helen*

* Corresponding author

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 522–530,
Baltimore, Maryland, USA, June 23-25 2014.

thought that her eldest son had the talent to play the violin, but the youngest son didn't.
Scenario C: Given sentence *Because of her neighbors' protests* as previous sentence, the negation focus should be *play the violin,* yielding interpretation *Helen didn't allow her youngest son to play the violin, but it didn't show whether he was allowed to do other things*.

In this paper, to well accommodate such contextual discourse information in negation focus identification, we propose a graph model to enrich normal intra-sentence features with various kinds of inter-sentence features from both lexical and topic perspectives. Besides, the standard PageRank algorithm is employed to optimize the graph model. Evaluation on the *SEM 2012 shared task corpus (Morante and Blanco, 2012) justifies our approach over several strong baselines.

The rest of this paper is organized as follows. Section 2 overviews the related work. Section 3 presents several strong baselines on negation focus identification with only intra-sentence features. Section 4 introduces our topic-driven word-based graph model with contextual discourse information. Section 5 reports the experimental results and analysis. Finally, we conclude our work in Section 6.

2 Related Work

Earlier studies of negation were almost in linguistics (e.g. Horn, 1989; van der Wouden, 1997), and there were only a few in natural language processing with focus on negation recognition in the biomedical domain. For example, Chapman et al. (2001) developed a rule-based negation recognition system, NegEx, to determine whether a finding mentioned within narrative medical reports is present or absent. Since the release of the BioScope corpus (Vincze et al., 2008), a freely available resource consisting of medical and biological texts, machine learning approaches begin to dominate the research on negation recognition (e.g. Morante et al., 2008; Li et al., 2010).

Generally, negation recognition includes three subtasks: cue detection, which detects and identifies possible negative expressions in a sentence, scope resolution, which determines the grammatical scope in a sentence affected by a negative expression, and focus identification, which identifies the constituent in a sentence most prominently or explicitly negated by a negative expression. This paper concentrates on the third subtask, negation focus identification.

Due to the increasing demand on deep understanding of natural language text, negation recognition has been drawing more and more attention in recent years, with a series of shared tasks and workshops, however, with focus on cue detection and scope resolution, such as the BioNLP 2009 shared task for negative event detection (Kim et al., 2009) and the ACL 2010 Workshop for scope resolution of negation and speculation (Morante and Sporleder, 2010), followed by a special issue of *Computational Linguistics* (Morante and Sporleder, 2012) for modality and negation.

The research on negation focus identification was pioneered by Blanco and Moldovan (2011), who investigated the negation phenomenon in semantic relations and proposed a supervised learning approach to identify the focus of a negation expression. However, although Morante and Blanco (2012) proposed negation focus identification as one of the *SEM'2012 shared tasks, only one team (Rosenberg and Bergler, 2012) [1] participated in this task. They identified negation focus using three kinds of heuristics and achieved 58.40 in F1-measure. This indicates great expectation in negation focus identification.

The key problem in current research on negation focus identification is its focus on intra-sentence information and large ignorance of inter-sentence information, which plays a critical role in the success of negation focus identification. For example, Ding (2011) made a qualitative analysis on implied negations in conversation and attempted to determine whether a sentence was negated by context information, from the linguistic perspective. Moreover, a negation focus is always associated with authors' intention in article. This indicates the great challenges in negation focus identification.

3 Baselines

Negation focus identification in *SEM'2012 shared tasks is restricted to verbal negations annotated with MNEG in PropBank, with only the constituent belonging to a semantic role selected as negation focus. Normally, a verbal negation expression (*not* or *n't*) is grammatically associated with its corresponding verb (e.g., *He did**n't** stop*). For details on annotation guidelines and

[1] In *SEM'2013, the shared task is changed with focus on "Semantic Textual Similarity".

examples for verbal negations, please refer to Blanco and Moldovan (2011).

For comparison, we choose the state-of-the-art system described in Blanco and Moldovan (2011), which employed various kinds of syntactic features and semantic role features, as one of our baselines. Since this system adopted C4.5 for training, we name it as *Baseline*$^{C4.5}$. In order to provide a stronger baseline, besides those features adopted in *Baseline*$^{C4.5}$, we added more refined intra-sentence features and adopted ranking Support Vector Machine (SVM) model for training. We name it as *Baseline*^{SVM}.

Following is a list of features adopted in the two baselines, for both Baseline$^{C4.5}$ and Baseline^{SVM},

- Basic features: first token and its part-of-speech (POS) tag of the focus candidate; the number of tokens in the focus candidate; relative position of the focus candidate among all the roles present in the sentence; negated verb and its POS tag of the negative expression;
- Syntactic features: the sequence of words from the beginning of the governing VP to the negated verb; the sequence of POS tags from the beginning of the governing VP to the negated verb; whether the governing VP contains a CC; whether the governing VP contains a RB.
- Semantic features: the syntactic label of semantic role A1; whether A1 contains POS tag DT, JJ, PRP, CD, RB, VB, and WP, as defined in Blanco and Moldovan (2011); whether A1 contains token any, anybody, anymore, anyone, anything, anytime, anywhere, certain, enough, full, many, much, other, some, specifics, too, and until, as defined in Blanco and Moldovan (2011); the syntactic label of the first semantic role in the sentence; the semantic label of the last semantic role in the sentence; the thematic role for A0/A1/A2/A3/A4 of the negated predicate.

and for *Baseline*^{SVM} only,

- Basic features: the named entity and its type in the focus candidate; relative position of the focus candidate to the negative expression (before or after).
- Syntactic features: the dependency path and its depth from the focus candidate to the negative expression; the constituent path and its depth from the focus candidate to the negative expression;

4 Exploring Contextual Discourse Information for Negation Focus Identification

While some of negation focuses could be identified by only intra-sentence information, others must be identified by contextual discourse information. Section 1 illustrates the necessity of such contextual discourse information in negation focus identification by giving three scenarios of different discourse contexts for negation expression ***n't*** in sentence (3).

For better illustration of the importance of contextual discourse information, Table 1 shows the statistics of intra- and inter-sentence information necessary for manual negation focus identification with 100 instances randomly extracted from the held-out dataset of *SEM'2012 shared task corpus. It shows that only 17 instances can be identified by intra-sentence information. It is surprising that inter-sentence information is indispensable in 77 instances, among which 42 instances need only inter-sentence information and 35 instances need both intra- and inter-sentence information. This indicates the great importance of contextual discourse information on negation focus identification. It is also interesting to note 6 instances are hard to determine even given both intra- and inter-sentence information.

Info	Number
#Intra-Sentence Only	17
#Inter-Sentence Only	42
#Both	35
#Hard to Identify	6

(Note: "Hard to Identify" means that it is hard for a human being to identify the negation focus even given both intra- and inter-sentence information.)

Table 1. Statistics of intra- and inter-sentence information on negation focus identification.

Statistically, we find that negation focus is always related with what authors repeatedly states in discourse context. This explains why contextual discourse information could help identify negation focus. While inter-sentence information provides the global characteristics from the discourse context perspective and intra-sentence information provides the local features from lexical, syntactic and semantic perspectives, both have their own contributions on negation focus identification.

In this paper, we first propose a graph model to gauge the importance of contextual discourse

information. Then, we incorporate both intra- and inter-sentence features into a machine learning-based framework for negation focus identification.

4.1 Graph Model

Graph models have been proven successful in many NLP applications, especially in representing the link relationships between words or sentences (Wan and Yang, 2008; Li et al., 2009). Generally, such models could construct a graph to compute the relevance between document theme and words.

In this paper, we propose a graph model to represent the contextual discourse information from both lexical and topic perspectives. In particular, a word-based graph model is proposed to represent the explicit relatedness among words in a discourse from the lexical perspective, while a topic-driven word-based model is proposed to enrich the implicit relatedness between words, by adding one more layer to the word-based graph model in representing the global topic distribution of the whole dataset. Besides, the PageRank algorithm (Page et al., 1998) is adopted to optimize the graph model.

Word-based Graph Model:

A word-based graph model can be defined as $G_{word}(W, E)$, where $W=\{w_i\}$ is the set of words in one document and $E=\{e_{ij}|w_i, w_j \in W\}$ is the set of directed edges between these words, as shown in Figure 1.

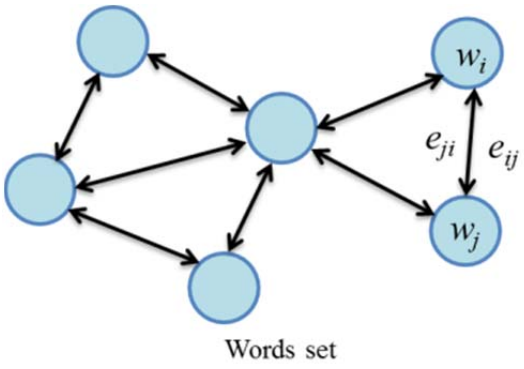


Figure 1. Word-based graph model.

In the word-based graph model, word node w_i is weighted to represent the correlation of the word with authors' intention. Since such correlation is more from the semantic perspective than the grammatical perspective, only content words are considered in our graph model, ignoring functional words (e.g., *the*, *to*,…). Especially, the content words limited to those with part-of-speech tags of JJ, NN, PRP, and VB. For simplicity, the weight of word node w_i is initialized to 1.

In addition, directed edge e_{ij} is weighted to represent the relatedness between word w_i and word w_j in a document with transition probability $P(j|i)$ from i to j, which is normalized as follows:

$$P(j|i) = \frac{Sim(w_i,w_j)}{\sum_k Sim(w_i,w_k)} \quad (1)$$

where k represents the nodes in discourse, and $Sim(w_i,w_j)$ denotes the similarity between w_i and w_j. In this paper, two kinds of information are used to calculate the similarity between words. One is word co-occurrence (if word w_i and word w_j occur in the same sentence or in the adjacent sentences, $Sim(w_i,w_j)$ increases 1), and the other is WordNet (Miller, 1995) based similarity. Please note that $Sim(w_i,w_i) = 0$ to avoid self-transition, and $Sim(w_i,w_j)$ and $Sim(w_j,w_i)$ may not be equal.

Finally, the weights of word nodes are calculated using the PageRank algorithm as follows:

$$Score^{(0)}(w_i) = 1$$
$$Score^{(n+1)}(w_i) = d\sum_{j\neq i} Score^{(n)}(w_j) \times P(j|i) + (1-d) \quad (2)$$

where d is the damping factor as in the PageRank algorithm.

Topic-driven Word-based Graph Model

While the above word-based graph model can well capture the relatedness between content words, it can only partially model the focus of a negation expression since negation focus is more directly related with topic than content. In order to reduce the gap, we propose a topic-driven word-based model by adding one more layer to refine the word-based graph model over the global topic distribution, as shown in Figure 2.

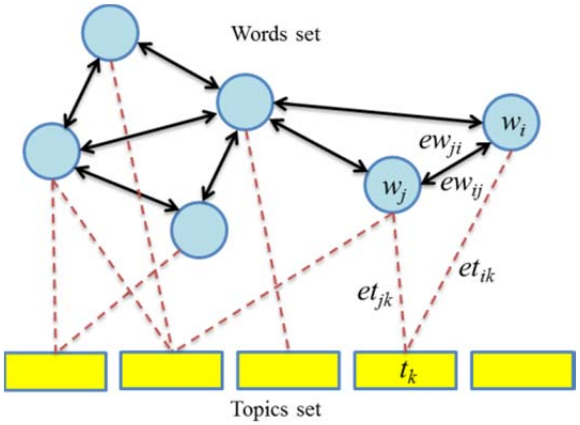


Figure 2. Topic-driven word-based graph model.

Here, the topics are extracted from all the documents in the *SEM 2012 shared task using the LDA Gibbs Sampling algorithm (Griffiths, 2002). In the topic-driven word-based graph model, the first layer denotes the relatedness among content words as captured in the above word-based graph model, and the second layer denotes the topic distribution, with the dashed lines between these two layers indicating the word-topic model return by LDA.

Formally, the topic-driven word-based two-layer graph is defined as G_{topic} (W, T, E_w, E_t), where $W=\{w_i\}$ is the set of words in one document and $T=\{t_i\}$ is the set of topics in all documents; $E_w=\{ew_{ij}|w_i, w_j \in W\}$ is the set of directed edges between words and $E_t=\{et_{ij}|w_i \in W, t_j \in T\}$ is the set of undirected edges between words and topics; transition probability $P_w(j|i)$ of ew_{ij} is defined as the same as $P(j|i)$ of the word-based graph model. Besides, transition probability $P_t(i,m)$ of et_{ij} in the word-topic model is defined as:

$$P_t(i,m) = \frac{Rel(w_i,t_m)}{\sum_k Rel(w_i,t_k)} \quad (3)$$

where $Rel(w_i, t_m)$ is the weight of word w_i in topic t_m calculated by the LDA Gibbs Sampling algorithm. On the basis, the transition probability $P_w(j|i)$ of ew_{ij} is updated by calculating as following:

$$P'_w(j|i) = \theta \cdot P_w(j|i) + (1-\theta) \cdot \frac{P_t(i,m) \times P_t(j,m)}{\sum_k P_t(i,k) \times P_t(j,k)} \quad (4)$$

where k represents all topics linked to both word w_i and word w_j, and $\theta \in [0,1]$ is the coefficient controlling the relative contributions from the lexical information in current document and the topic information in all documents.

Finally, the weights of word nodes are calculated using the PageRank algorithm as follows:

$$Score^{(0)}(w_i) = 1$$
$$Score^{(n+1)}(w_i) = d \sum_{j \neq i} Score^{(n)}(w_j) \times P'_w(j|i) + (1-d) \quad (5)$$

where d is the damping factor as in the PageRank algorithm.

4.2 Negation Focus Identification via Graph Model

Given the graph models and the PageRank optimization algorithm discussed above, four kinds of contextual discourse information are extracted as inter-sentence features (Table 2).

In particular, the total weight and the max weight of words in the focus candidate are calculated as follows:

$$Weight_{total} = \sum_i Score^{(final)}(w_i) \quad (6)$$

$$Weight_{max} = \max_i Score^{(final)}(w_i) \quad (7)$$

where i represents the content words in the focus candidate. These two kinds of weights focus on different aspects about the focus candidate with the former on the contribution of content words, which is more beneficial for a long focus candidate, and the latter biased towards the focus candidate which contains some critical word in a discourse.

No	Feature
1	Total weight of words in the focus candidate using the co-occurrence similarity.
2	Max weight of words in the focus candidate using the co-occurrence similarity.
3	Total weight of words in the focus candidate using the WordNet similarity.
4	Max weight of words in the focus candidate using the WordNet similarity.

Table 2. Inter-sentence features extracted from graph model.

For evaluating the contribution of contextual discourse information on negation focus identification directly, we incorporate the four inter-sentence features from the topic-driven word-based graph model into a negation focus identifier.

5 Experimentation

In this section, we describe experimental settings and systematically evaluate our negation focus identification approach with focus on exploring the effectiveness of contextual discourse information.

5.1 Experimental Settings

Dataset

In all our experiments, we employ the *SEM'2012 shared task corpus (Morante and Blanco, 2012)[2]. As a freely downloadable resource, the *SEM shared task corpus is annotated on top of PropBank, which uses the WSJ section of the Penn TreeBank. In particular, negation focus annotation on this corpus is restricted to verbal negations (with corresponding mark

[2] http://www.clips.ua.ac.be/sem2012-st-neg/

MNEG in PropBank). On 50% of the corpus annotated by two annotators, the inter-annotator agreement was 0.72 (Blanco and Moldovan, 2011). Along with negation focus annotation, this corpus also contains other annotations, such as POS tag, named entity, chunk, constituent tree, dependency tree, and semantic role.

In total, this corpus provides 3,544 instances of negation focus annotations. For fair comparison, we adopt the same partition as *SEM'2012 shared task in all our experiments, i.e., with 2,302 for training, 530 for development, and 712 for testing. Although for each instance, the corpus only provides the current sentence, the previous and next sentences as its context, we sort to the Penn TreeBank[3] to obtain the corresponding document as its discourse context.

Evaluation Metrics

Same as the *SEM'2012 shared task, the evaluation is made using precision, recall, and F1-score. Especially, a true positive (TP) requires an exact match for the negation focus, a false positive (FP) occurs when a system predicts a non-existing negation focus, and a false negative (FN) occurs when the gold annotations specify a negation focus but the system makes no prediction. For each instance, the predicted focus is considered correct if it is a complete match with a gold annotation.

Beside, to show whether an improvement is significant, we conducted significance testing using z-test, as described in Blanco and Moldovan (2011).

Toolkits

In our experiments, we report not only the default performance with gold additional annotated features provided by the *SEM'2012 shared task corpus and the Penn TreeBank, but also the performance with various kinds of features extracted automatically, using following toolkits:

- Syntactic Parser: We employ the Stanford Parser[4] (Klein and Manning, 2003; De Marneffe et al., 2006) for tokenization, constituent and dependency parsing.
- Named Entity Recognizer: We employ the Stanford NER[5] (Finkel et al., 2005) to obtain named entities.
- Semantic Role Labeler: We employ the semantic role labeler, as described in Punyakanok et al (2008).
- Topic Modeler: For estimating transition probability Pt(i,m), we employ GibbsLDA++[6], an LDA model using Gibbs Sampling technique for parameter estimation and inference.
- Classifier: We employ SVM^{Light} [7] with default parameters as our classifier.

5.2 Experimental Results

With Only Intra-sentence Information

Table 3 shows the performance of the two baselines, the decision tree-based classifier as in Blanco and Moldovan (2011) and our ranking SVM-based classifier. It shows that our ranking SVM-based baseline slightly improves the F1-measure by 2.52% over the decision tree-based baseline, largely due to the incorporation of more refined features.

System	P(%)	R(%)	F1
Baseline$^{C4.5}$	66.73	49.93	57.12
Baseline^{SVM}	60.22	59.07	59.64

Table 3. Performance of baselines with only intra-sentence information.

Error analysis of the ranking SVM-based baseline on development data shows that 72% of them are caused by the ignorance of inter-sentence information. For example, among the 42 instances listed in the category of "#Inter-Sentence Only" in Table 1, only 7 instances can be identified correctly by the ranking SVM-based classifier. With about 4 focus candidates in one sentence on average, this percentage is even lower than random.

With Only Inter-sentence Information

For exploring the usefulness of pure contextual discourse information in negation focus identification, we only employ inter-sentence features into ranking SVM-based classifier. First of all, we estimate two parameters for our topic-driven word-based graph model: topic number T for topic model and coefficient θ between $P_w(j|i)$ and $P_t(i,m)$ in Formula 4.

Given the LDA Gibbs Sampling model with parameters $\alpha = 50/T$ and $\beta = 0.1$, we vary T from 20 to 100 with an interval of 10 to find the opti-

[3] http://www.cis.upenn.edu/~treebank/
[4] http://nlp.stanford.edu/software/lex-parser.shtml
[5] http://nlp.stanford.edu/ner/
[6] http://gibbslda.sourceforge.net/
[7] http://svmlight.joachims.org

mal T. Figure 3 shows the experiment results of varying T (with $\theta = 0.5$) on development data. It shows that the best performance is achieved when T = 50 with 51.11 in F1). Therefore, we set T as 50 in our following experiments.

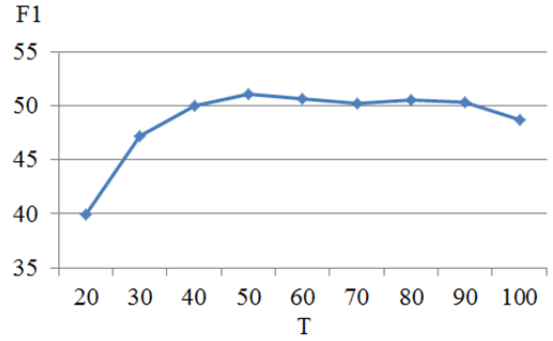


Figure 3. Performance with varying T.

For parameter θ, a trade-off between the transition probability $P_w(j|i)$ (word to word) and the transition probability $P_t(i,m)$ (word and topic) to update $P'_w(j|i)$, we vary it from 0 to 1 with an interval of 0.1. Figure 4 shows the experiment results of varying θ (with T=50) on development data. It shows that the best performance is achieved when $\theta = 0.6$, which are adopted hereafter in all our experiments. This indicates that direct lexical information in current document contributes more than indirect topic information in all documents on negation focus identification. It also shows that direct lexical information in current document and indirect topic information in all documents are much complementary on negation focus identification.

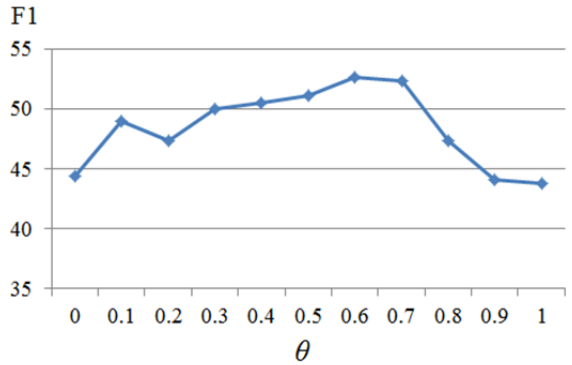


Figure 4. Performance with varying θ.

System	P(%)	R(%)	F1
using word-based graph model	45.62	42.02	43.75
using topic-driven word-based graph model	54.59	50.76	52.61

Table 4. Performance with only inter-sentence information.

Table 4 shows the performance of negation focus identification with only inter-sentence features. It also shows that the system with inter-sentence features from the topic-driven word-based graph model significantly improves the F1-measure by 8.86 over the system with inter-sentence features from the word-based graph model, largely due to the usefulness of topic information.

In comparison with Table 3, it shows that the system with only intra-sentence features achieves better performance than the one with only inter-sentence features (59.64 vs. 52.61 in F1-measure).

With both Intra- and Inter-sentence Information

Table 5 shows that enriching intra-sentence features with inter-sentence features significantly ($p<0.01$) improve the performance by 9.85 in F1-measure than the better baseline. This indicates the usefulness of such contextual discourse information and the effectiveness of our topic-driven word-based graph model in negation focus identification.

System	P(%)	R(%)	F1
Baseline$^{C4.5}$ with intra feat. only	66.73	49.93	57.12
Baseline^{SVM} with intra feat. only	60.22	59.07	59.64
Ours with Both feat. using word-based GM	64.93	62.47	63.68
Ours with Both feat. using topic-driven word-based GM	71.67	67.43	69.49

(Note: "feat." denotes features; "GM" denotes graph model.)

Table 5. Performance comparison of systems on negation focus identification.

System	P(%)	R(%)	F1
Baseline$^{C4.5}$ with intra feat. only (auto)	60.94	44.62	51.52
Baseline^{SVM} with intra feat. Only (auto)	53.81	51.67	52.72
Ours with Both feat. using word-based GM (auto)	58.77	57.19	57.97
Ours with Both feat. using topic-driven word-based GM (auto)	66.74	64.53	65.62

Table 6. Performance comparison of systems on negation focus identification with automatically extracted features.

Besides, Table 6 shows the performance of our best system with all features automatically extracted using the toolkits as described in Section 5.1. Compared with our best system employing gold additional annotated features (the last line in Table 5), the homologous system with automatically extracted features (the last line in Table 6) only decrease of less than 4 in F1-measure. This demonstrates the achievability of our approach.

In comparison with the best-reported performance on the *SEM'2012 shared task (Rosenberg and Bergler, 2012), our system performs better by about 11 in F-measure.

5.3 Discussion

While this paper verifies the usefulness of contextual discourse information on negation focus identification, the performance with only inter-sentence features is still weaker than that with only intra-sentence features. There are two main reasons. On the one hand, the former employs an unsupervised approach without prior knowledge for training. On the other hand, the usefulness of inter-sentence features depends on the assumption that a negation focus relates to the meaning of which is most relevant to authors' intention in a discourse. If there lacks relevant information in a discourse context, negation focus will become difficult to be identified only by inter-sentence features.

Error analysis also shows that some of the negation focuses are very difficult to be identified, even for a human being. Consider the sentence (3) in Section 1, if given sentence *because of her neighbors' protests, but her husband doesn't think so* as its following context, both *Helen* and *to play the violin* can become the negation focus. Moreover, the inter-annotator agreement in the first round of negation focus annotation can only reach 0.72 (Blanco and Moldovan, 2011). This indicates inherent difficulty in negation focus identification.

6 Conclusion

In this paper, we propose a graph model to enrich intra-sentence features with inter-sentence features from both lexical and topic perspectives. In this graph model, the relatedness between words is calculated by word co-occurrence, WordNet-based similarity, and topic-driven similarity. Evaluation on the *SEM 2012 shared task corpus indicates the usefulness of contextual discourse information on negation focus identification and our graph model in capturing such global information.

In future work, we will focus on exploring more contextual discourse information via the graph model and better ways of integrating intra- and inter-sentence information on negation focus identification.

Acknowledgments

This research is supported by the National Natural Science Foundation of China, No.61272260, No.61331011, No.61273320, the Natural Science Foundation of Jiangsu Province, No. BK2011282, the Major Project of College Natural Science Foundation of Jiangsu Province, No.11KIJ520003, and the Graduates Project of Science and Innovation, No. CXZZ12_0818. The authors would like to thank the anonymous reviewers for their insightful comments and suggestions. Our sincere thanks are also extended to Dr. Zhongqing Wang for his valuable discussions during this study.

Reference

Eduardo Blanco and Dan Moldovan. 2011. Semantic Representation of Negation Using Focus Detection. *In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 581-589, Portland, Oregon, June 19-24, 2011.

Wendy W. Chapman, Will Bridewell, Paul Hanbury, Gregory F. Cooper, and Bruce G. Buchanan. 2001. A simple algorithm for identifying negated findings and diseases in discharge summaries. *Journal of Biomedical Informatics*, 34:301-310.

Marie-Catherine De Marneffe, Bill MacCartney and Christopher D. Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. *In Proceedings of LREC'2006*.

Yun Ding. 2011. Implied Negation in Discourse. *Journal of Theory and Practice in Language Studies*, 1(1): 44-51, Jan 2011.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. *In Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363-370, Stroudsburg, PA, USA.

Tom Griffiths. 2002. Gibbs sampling in the generative model of Latent Dirichlet Allocation. Tech. rep., Stanford University.

Laurence R Horn. 1989. A Natural History of Negation. Chicago University Press, Chicago, IL.

Fangtao Li, Yang Tang, Minlie Huang, and Xiaoyan Zhu. 2009. Answering Opinion Questions with Random Walks on Graphs. *In Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, pages 737-745, Suntec, Singapore, 2-7 Aug 2009.

Junhui Li, Guodong Zhou, Hongling Wang, and Qiaoming Zhu. 2010. Learning the Scope of Negation via Shallow Semantic Parsing. *In Proceedings of the 23rd International Conference on Computational Linguistics*. Stroudsburg, PA, USA: Association for Computational Linguistics, 671-679.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of BioNLP'09 Shared Task on Event Extraction. *In Proceedings of the BioNLP'2009 Workshop Companion Volume for Shared Task*. Stroudsburg, PA, USA: Association for Computational Linguistics, 1-9.

Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. *In Proceedings of the 41st Meeting of the Association for Computational Linguistics*, pages 423-430.

George A. Miller. 1995. Wordnet: a lexical database for english. Commun. ACM, 38(11):39-41.

Roser Morante, Anthony Liekens and Walter Daelemans. 2008. Learning the Scope of Negation in Biomedical Texts. *In Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 715-724, Honolulu, October 2008.

Roser Morante and Caroline Sporleder, editors. 2010. *In Proceedings of the Workshop on Negation and Speculation in Natural Language Processing*. University of Antwerp, Uppsala, Sweden.

Roser Morante and Eduardo Blanco. 2012. *SEM 2012 Shared Task: Resolving the Scope and Focus of Negation. *In Proceedings of the First Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 265-274, Montreal, Canada, June 7-8, 2012.

Roser Morante and Caroline Sporleder. 2012. Modality and Negation: An Introduction to the Special Issue. *Computational Linguistics*, 2012, 38(2): 223-260.

Roser Morante and Walter Daelemans. 2012. Conan Doyle-neg: Annotation of negation cues and their scope in Conan Doyle stories. *In Proceedings of LREC 2012*, Istambul.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1998. The pagerank citation ranking: Bringing order to the web. *Technical report*, Stanford University.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257-287, June.

Sabine Rosenberg and Sabine Bergler. 2012. UConcordia: CLaC Negation Focus Detection at *Sem 2012. *In Proceedings of the First Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 294-300, Montreal, Canada, June 7-8, 2012.

Ton van der Wouden. 1997. Negative Contexts: Collocation, Polarity, and Multiple Negation. Routledge, London.

Veronika Vincze, György Szarvas, Richárd Farkas, György Móra, and János Csirik. 2008. The BioScope corpus: biomedical texts annotated for uncertainty,negation and their scopes. *BMC Bioinformatics*, 9(Suppl 11):S9.

Xiaojun Wan and Jianwu Yang. 2008. Multi-document summarization using cluster-based link analysis. *In Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 299-306.

New Word Detection for Sentiment Analysis

Minlie Huang, Borui Ye*, Yichen Wang, Haiqiang Chen, Junjun Cheng**, Xiaoyan Zhu**
State Key Lab. of Intelligent Technology and Systems, National Lab. for Information Science and Technology, Dept. of Computer Science and Technology, Tsinghua University, Beijing 100084, PR China
*Dept. of Communication Engineering, Beijing University of Posts and Telecommunications
**China Information Technology Security Evaluation Center
aihuang@tsinghua.edu.cn

Abstract

Automatic extraction of new words is an indispensable precursor to many NLP tasks such as Chinese word segmentation, named entity extraction, and sentiment analysis. This paper aims at extracting *new sentiment words* from large-scale user-generated content. We propose a fully unsupervised, purely data-driven framework for this purpose. We design statistical measures respectively to quantify the utility of a lexical pattern and to measure the possibility of a word being a new word. The method is almost free of linguistic resources (except POS tags), and requires no elaborated linguistic rules. We also demonstrate how new sentiment word will benefit sentiment analysis. Experiment results demonstrate the effectiveness of the proposed method.

1 Introduction

New words on the Internet have been emerging all the time, particularly in user-generated content. Users like to update and share their information on social websites with their own language styles, among which new political/social/cultural words are constantly used.

However, such new words have made many natural language processing tasks more challenging. Automatic extraction of new words is indispensable to many tasks such as Chinese word segmentation, machine translation, named entity extraction, question answering, and sentiment analysis. New word detection is one of the most critical issues in Chinese word segmentation. Recent studies (Sproat and Emerson, 2003) (Chen, 2003) have shown that more than 60% of word segmentation errors result from new words. Statistics show that more than 1000 new Chinese words appear every year (Thesaurus Research Center, 2003). These words are mostly domain-specific technical terms and time-sensitive political/social /cultural terms. Most of them are not yet correctly recognized by the segmentation algorithm, and remain as out of vocabulary (OOV) words.

New word detection is also important for sentiment analysis such as opinionated phrase extraction and polarity classification. A sentiment phrase with complete meaning should have a correct boundary, however, characters in a new word may be broken up. For example, in a sentence " 表演/n 非常/adv 给/v 力/n（artists' performance is very impressive）" the two Chinese characters "给/v 力/n(cool; powerful)" should always be extracted together. In polarity classification, new words can be informative features for classification models. In the previous example, "给力(cool; powerful)" is a strong feature for classification models while each single character is not. Adding new words as feature in classification models will improve the performance of polarity classification, as demonstrated later in this paper.

This paper aims to detect new word for sentiment analysis. We are particulary interested in extracting *new sentiment word* that can express opinions or sentiment, which is of high value towards sentiment analysis. *New sentiment word*, as exemplified in Table 1, is a sub-class of multi-word expressions which is a sequence of neighboring words *"whose exact and unambiguous meaning or connotation cannot be derived from the meaning or connotation of its components"* (Choueka, 1988). Such new words cannot be directly identified using grammatical rules, which poses a major challenge to automatic analysis. Moreover, existing lexical resources never have adequate and timely coverage since new words appear constantly. People thus resort to statistical methods such as Pointwise Mutual Information (Church and Hanks, 1990), Symmetrical Conditional Probability

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 531–541,
Baltimore, Maryland, USA, June 23-25 2014.

(da Silva and Lopes, 1999), Mutual Expectation (Dias et al., 2000), Enhanced Mutual Information (Zhang et al., 2009), and Multi-word Expression Distance (Bu et al., 2010).

New word	English Translation	Polarity
口爱	lovely	positive
杯具	tragic/tragedy	negative
给力	very cool; powerful	positive
坑爹	reverse one's expectation	negative

Table 1: Examples of new sentiment word.

Our central idea for new sentiment word detection is as follows: Starting from very few seed words (for example, just one seed word), we can extract lexical patterns that have strong statistical association with the seed words; the extracted lexical patterns can be further used in finding more new words, and the most probable new words can be added into the seed word set for the next iteration; and the process can be run iteratively until a stop condition is met. The key issues are to measure the utility of a pattern and to quantify the possibility of a word being a new word. The main contributions of this paper are summarized as follows:

- We propose a novel framework for new word detection from large-scale user-generated data. This framework is fully unsupervised and purely data-driven, and requires very lightweight linguistic resources (i.e., only POS tags).
- We design statistical measures to quantify the utility of a pattern and to quantify the possibility of a word being a new word, respectively. No elaborated linguistic rules are needed to filter undesirable results. This feature may enable our approach to be portable to other languages.
- We investigate the problem of polarity prediction of new sentiment word and demonstrate that inclusion of new sentiment word benefits sentiment classification tasks.

The rest of the paper is structured as follows: we will introduce related work in the next section. We will describe the proposed method in Section 3, including definitions, the overview of the algorithm, and the statistical measures for addressing the two key issues. We then present the experiments in Section 4. Finally, the work is summarized in Section 5.

2 Related Work

New word detection has been usually interweaved with word segmentation, particularly in Chinese NLP. In these works, new word detection is considered as an integral part of segmentation, where new words are identified as the most probable segments inferred by the probabilistic models; and the detected new word can be further used to improve word segmentation. Typical models include conditional random fields proposed by (Peng et al., 2004), and a joint model trained with adaptive online gradient descent based on feature frequency information (Sun et al., 2012).

Another line is to treat new word detection as a separate task, usually preceded by part-of-speech tagging. The first genre of such studies is to leverage complex linguistic rules or knowledge. For example, Justeson and Katz (1995) extracted technical terminologies from documents using a regular expression. Argamon et al. (1998) segmented the POS sequence of a multi-word into small POS tiles, counted tile frequency in the new word and non-new-word on the training set respectively, and detected new words using these counts. Chen and Ma (2002) employed morphological and statistical rules to extract Chinese new word. The second genre of the studies is to treat new word detection as a classification problem. Zhou (2005) proposed a discriminative Markov Model to detect new words by chunking one or more separated words. In (Li et al., 2005), new word detection was viewed as a binary classification problem. However, these supervised models requires not only heavy engineering of linguistic features, but also expensive annotation of training data.

User behavior data has recently been explored for finding new words. Zheng et al. (2009) explored user typing behaviors in Sogou Chinese Pinyin input method to detect new words. Zhang et al. (2010) proposed to use dynamic time warping to detect new words from query logs. However, both of the work are limited due to the public unavailability of expensive commercial resources.

Statistical methods for new word detection have been extensively studied, and in some sense exhibit advantages over linguistics-based methods. In this setting, new word detection is mostly

known as multi-word expression extraction. To measure multi-word association, the first model is Pointwise Mutual Information (PMI) (Church and Hanks, 1990). Since then, a variety of statistical methods have been proposed to measure bi-gram association, such as Log-likelihood (Dunning, 1993) and Symmetrical Conditional Probability (SCP) (da Silva and Lopes, 1999). Among all the 84 bi-gram association measures, PMI has been reported to be the best one in Czech data (Pecina, 2005). In order to measure arbitrary n-grams, most common strategies are to separate n-gram into two parts X and Y so that existing bi-gram methods can be used (da Silva and Lopes, 1999; Dias et al., 2000; Schone and Jurafsky, 2001). Zhang et al. (2009) proposed Enhanced Mutual Information (EMI) which measures the cohesion of n-gram by the frequency of itself and the frequency of each single word. Based on the information distance theory, Bu et al. (2010) proposed multi-word expression distance (MED) and the normalized version, and reported superior performance to EMI, SCP, and other measures.

3 Methodology

3.1 Definitions

Definition 3.1 (**Adverbial word**). Words that are used mainly to modify a verb or an adjective, such as "太(too)", "非常(very)", "十分(very)", and "特别(specially)".

Definition 3.2 (**Auxiliary word**). Words that are auxiliaries, model particles, or punctuation marks. In Chinese, such words are like "着,了,啦,的,啊", and punctuation marks include "，。！？；： " and so on.

Definition 3.3 (**Lexical Pattern**). A lexical pattern is a triplet $< AD, *, AU >$, where AD is an adverbial word, the wildcard $*$ means an arbitrary number of words [1], and AU denotes an auxiliary word.

Table 2 gives some examples of lexical patterns. In order to obtain lexical patterns, we can define regular expressions with POS tags [2] and apply the regular expressions on POS tagged texts. Since the tags of adverbial and auxiliary words are relatively static and can be easily identified, such a method can safely obtain lexical patterns.

Pattern	Frequency
<"都",*,"了">	562,057
<"都",*,"的">	387,649
<"太",*,"了">	380,470
<"不",*,"， ">	369,702

Table 2: Examples of lexical pattern. The frequency is counted on 237,108,977 Weibo posts.

3.2 The Algorithm Overview

The algorithm works as follows: starting from very few seed words (for example, a word in Table 1), the algorithm can find lexical patterns that have strong statistical association with the seed words in which the likelihood ratio test (LRT) is used to quantify the degree of association. Subsequently, the extracted lexical patterns can be further used in finding more new words. We design several measures to quantify the possibility of a candidate word being a new word, and the top-ranked words will be added into the seed word set for the next iteration. The process can be run iteratively until a stop condition is met. Note that we do not augment the pattern set ($\mathcal{P}$) at each iteration, instead, we keep a fixed small number of patterns during iteration because this strategy produces optimal results.

From linguistic perspectives, new sentiment words are commonly modified by adverbial words and thus can be extracted by lexical patterns. This is the reason why the algorithm will work. Our algorithm is in spirit to double propagation (Qiu et al., 2011), however, the differences are apparent in that: firstly, we use very lightweight linguistic information (except POS tags); secondly, our major contributions are to propose statistical measures to address the following key issues: first, to measure the utility of lexical patterns; second, to measure the possibility of a candidate word being a new word.

3.3 Measuring the Utility of a Pattern

The first key issue is to quantify the utility of a pattern at each iteration. This can be measured by the association of a pattern to the current word set used in the algorithm. The likelihood ratio tests (Dunning, 1993) is used for this purpose. This association model has also been used to model association between opinion target words by (Hai et

[1] We set the number to 3 words in this work considering computation costs.

[2] Such expressions are very simple and easy to write because we only need to consider POS tags of adverbial and auxiliary word.

Algorithm 1: New word detection algorithm

Input:
$\mathcal{D}$: a large set of POS tagged posts
$\mathcal{W}_s$: a set of seed words
k_p: the number of patterns chosen at each iteration
k_c: the number of patterns in the candidate pattern set
k_w: the number of words added at each iteration
K: the number of words returned
Output: A list of ranked new words $\mathcal{W}$

Obtain all lexical patterns using regular expressions on $\mathcal{D}$;
Count the frequency of each lexical pattern and extract words matched by each pattern ;
Obtain top k_c frequent patterns as candidate pattern set $\mathcal{P}_c$ and top 5,000 frequent words as candidate word set $\mathcal{W}_c$;
$\mathcal{P} = \Phi$; $\mathcal{W}$=$\mathcal{W}_s$; t = 0 ;
for $|\mathcal{W}| < K$ **do**
    Use $\mathcal{W}$ to score each pattern in $\mathcal{P}_c$ with $U(p)$;
    $\mathcal{P} = \{top\ k_p\ patterns\}$;
    Use $\mathcal{P}$ to extract new words and if the words are in $\mathcal{W}_c$, score them with $F(w)$;
    $\mathcal{W} = \mathcal{W} \bigcup \{top\ k_w\ words\}$;
    $\mathcal{W}_c$ = $\mathcal{W}_c$ - $\mathcal{W}$;
Sort words in $\mathcal{W}$ with $F(w)$;
Output the ranked list of words in $\mathcal{W}$;

al., 2012).

The LRT is well known for not relying critically on the assumption of normality, instead, it uses the asymptotic assumption of the generalized likelihood ratio. In practice, the use of likelihood ratios tends to result in significant improvements in text-analysis performance.

In our problem, LRT computes a contingency table of a pattern p and a word w, derived from the corpus statistics, as given in Table 3, where $k_1(w, p)$ is the number of documents that w matches pattern p, $k_2(w, \bar{p})$ is the number of documents that w occurs while p does not, $k_3(\bar{w}, p)$ is the number of documents that p occurs while w does not, and $k_4(\bar{w}, \bar{p})$ is the number of documents containing neither p nor w.

Statistics	p	$\bar{p}$
w	$k_1(w, p)$	$k_2(w, \bar{p})$
$\bar{w}$	$k_3(\bar{w}, p)$	$k_4(\bar{w}, \bar{p})$

Table 3: Contingency table for likelihood ratio test (LRT).

Based on the statistics shown in Table 3, the likelihood ratio tests (LRT) model captures the statistical association between a pattern p and a word w by employing the following formula:

$$LRT(p, w) = log \frac{L(\rho_1, k_1, n_1) * L(\rho_2, k_2, n_2)}{L(\rho, k_1, n_1) * L(\rho, k_2, n_2)} \quad (1)$$

where:
$L(\rho, k, n) = \rho^k * (1 - \rho)^{n-k}$; $n_1 = k_1 + k_3$; $n_2 = k_2 + k_4$; $\rho_1 = k_1/n_1$; $\rho_2 = k_2/n_2$; $\rho = (k_1 + k_2)/(n_1 + n_2)$.

Thus, the utility of a pattern can be measured as follows:

$$U(p) = \sum_{w_i \in \mathcal{W}} LRT(p, w_i) \quad (2)$$

where $\mathcal{W}$ is the current word set used in the algorithm (see Algorithm 1).

3.4 Measuring the Possibility of Being New Words

Another key issue in the proposed algorithm is to quantify the possibility of a candidate word being a new word. We consider several factors for this purpose.

3.4.1 Likelihood Ratio Test

Very similar to the pattern utility measure, LRT can also be used to measure the association of a candidate word to a given pattern set, as follows:

$$LRT(w) = \sum_{p_i \in \mathcal{P}} LRT(w, p_i) \quad (3)$$

where $\mathcal{P}$ is the current pattern set used in the algorithm (see Algorithm 1), and p_i is a lexical pattern.

This measure only quantifies the association of a candidate word to the given pattern set. It tells nothing about the possibility of a word being a new word, however, a new *sentiment* word, should have close association with the lexical patterns. This has linguistic interpretations because new sentiment words are commonly modified by adverbial words and thus should have close association with lexical patterns. This measure is proved to be an influential factor by our experiments in Section 4.3.

3.4.2 Left Pattern Entropy

If a candidate word is a new word, it will be more commonly used with diversified lexical patterns since the non-compositionality of new word means that the word can be used in many different linguistic scenarios. This can be measured by information entropy, as follows:

$$LPE(w) = -\sum_{l_i \in L(\mathcal{P}_c, w)} \frac{c(l_i, w)}{N(w)} * log \frac{c(l_i, w)}{N(w)} \quad (4)$$

where $L(\mathcal{P}_c, w)$ is the set of left word of all patterns by which word w can be matched in $\mathcal{P}_c$, $c(l_i, w)$ is the count that word w can be matched by patterns whose left word is l_i, and $N(w)$ is the count that word w can be matched by the patterns in $\mathcal{P}_c$. Note that we use $\mathcal{P}_c$, instead of $\mathcal{P}$, because the latter set is very small while computing entropy needs a large number of patterns. Tuning the size of $\mathcal{P}_c$ will be further discussed in Section 4.4.

3.4.3 New Word Probability

Some words occur very frequently and can be widely matched by lexical patterns, but they are not new words. For example, " 爱吃(love to eat)" and " 爱说(love to talk)" can be matched by many lexical patterns, however, they are not new words due to the lack of non-compositionality. In such words, each single character has high probability to be a word. Thus, we design the following measure to favor this observation.

$$NWP(w) = \prod_{i=1}^{n} \frac{p(w_i)}{1 - p(w_i)} \quad (5)$$

where $w = w_1 w_2 \dots w_n$, each w_i is a single character, and $p(w_i)$ is the probability of the character w_i being a word, as computed as follows:

$$p(w_i) = \frac{all(w_i) - s(w_i)}{all(w_i)}$$

where $all(w_i)$ is the total frequency of w_i, and $s(w_i)$ is the frequency of w_i being a single character word. Obviously, in order to obtain the value of $s(w_i)$, some particular Chinese word segmentation tool is required. In this work, we resort to ICTCLAS (Zhang et al., 2003), a widely used tool in the literature.

3.4.4 Non-compositionality Measures

New words are usually multi-word expressions, where a variety of statistical measures have been proposed to detect multi-word expressions. Thus, such measures can be naturally incorporated into our algorithm.

The first measure is enhanced mutual information (EMI) (Zhang et al., 2009):

$$EMI(w) = log_2 \frac{F/N}{\prod_{i=1}^{n} \frac{F_i - F}{N}} \quad (6)$$

where F is the number of posts in which a multi-word expression $w = w_1 w_2 \dots w_n$ occurs, F_i is the number of posts where w_i occurs, and N is the total number of posts. The key idea of EMI is to measure word pair' s dependency as the ratio of its probability of being a multi-word to its probability of not being a multi-word. The larger the value, the more possible the expression will be a multi-word expression.

The second measure we take into account is normalized multi-word expression distance (Bu et al., 2010), which has been proposed to measure the non-compositionality of multi-word expressions.

$$NMED(w) = \frac{log|\mu(w)| - log|\phi(w)|}{logN - log|\phi(w)|} \quad (7)$$

where $\mu(w)$ is the set of documents in which all single words in $w = w_1 w_2 \dots w_n$ co-occur, $\phi(w)$ is the set of documents in which word w occurs as a whole, and N is the total number of documents. Different from EMI, this measure is a strict distance metric, meaning that a smaller value indicates a larger possibility of being a multi-word expression. As can be seen from the formula, the key idea of this metric is to compute the ratio of the co-occurrence of all words in a multi-word expressions to the occurrence of the whole expression.

3.4.5 Configurations to Combine Various Factors

Taking into account the aforementioned factors, we have different settings to score a new word, as follows:

$$F_{LRT}(w) = LRT(w) \quad (8)$$

$$F_{LPE}(w) = LRT(w) * LPE(w) \quad (9)$$

$$F_{NWP}(w) = LRT(w) * LPE(w) * NWP(w) \quad (10)$$

$$F_{EMI}(w) = LRT(w) * LPE(w) * EMI(w) \quad (11)$$

$$F_{NMED}(w) = \frac{LRT(w) * LPE(w)}{NMED(w)} \quad (12)$$

4 Experiment

In this section, we will conduct the following experiments: first, we will compare our method to several baselines, and perform parameter tuning with extensive experiments; second, we will classify polarity of new sentiment words using two methods; third, we will demonstrate how new sentiment words will benefit sentiment classification.

4.1 Data Preparation

We crawled 237,108,977 Weibo posts from http://www.weibo.com, the largest social website in China. These posts range from January of 2011 to December of 2012. The posts were then part-of-speech tagged using a Chinese word segmentation tool named ICTCLAS (Zhang et al., 2003).

Then, we asked two annotators to label the top 5,000 frequent words that were extracted by lexical patterns as described in Algorithm 1. The annotators were requested to judge whether a candidate word is a new word, and also to judge the polarity of a new word (positive, negative, and neutral). If there is a disagreement on either of the two tasks, discussions are required to make the final decision. The annotation led to 323 new words, among which there are 116 positive words, 112 negative words, and 95 neutral words[3].

4.2 Evaluation Metric

As our algorithm outputs a ranked list of words, we adapt average precision to evaluate the performance of new sentiment word detection. The metric is computed as follows:

$$AP(K) = \frac{\sum_{k=1}^{K} P(k) * rel(k)}{\sum_{k=1}^{K} rel(k)}$$

where $P(k)$ is the precision at cut-off k, $rel(k)$ is 1 if the word at position k is a new word and 0 otherwise, and K is the number of words in the ranked list. A perfect list (all top K items are correct) has an AP value of 1.0.

4.3 Evaluation of Different Measures and Comparison to Baselines

First, we assess the influence of likelihood ratio test, which measures the association of a word to the pattern set. As can be seen from Table 4, the association model (LRT) remarkably boosts the performance of new word detection, indicating LRT is a key factor for new sentiment word extraction. From linguistic perspectives, new sentiment words are commonly modified by adverbial words and thus should have close association with lexical patterns.

Second, we compare different settings of our method to two baselines. The first one is enhanced mutual information (EMI) where we set $F(w) = EMI(w)$ (Zhang et al., 2009) and the second baseline is normalized multi-word expression distance (NMED) (Bu et al., 2010) where we set $F(w) = NMED(w)$. The results are shown in Figure 1. As can be seen, all the proposed measures outperform the two baselines (EMI and $NMED$) remarkably and consistently. The setting of F_{NMED} produces the best performance. Adding $NMED$ or EMI leads to remarkable improvements because of their capability of measuring non-compositionality of new words. Only using LRT can obtain a fairly good results when K is small, however, the performance drops sharply because it's unable to measure non-compositionality. Comparison between $LRT + LPE$ (or $LRT + LPE + NWP$) and LRT shows that inclusion of left pattern entropy also boosts the performance apparently. However, the new word probability (NWP) has only marginal contribution to improvement.

In the above experiments, we set $k_p = 5$ (the number of patterns chosen at each iteration) and $k_w = 10$ (the number of words added at each iteration), which is the optimal setting and will be discussed in the next subsection. And only one seed word "坑爹(reverse one's expectation)" is used.

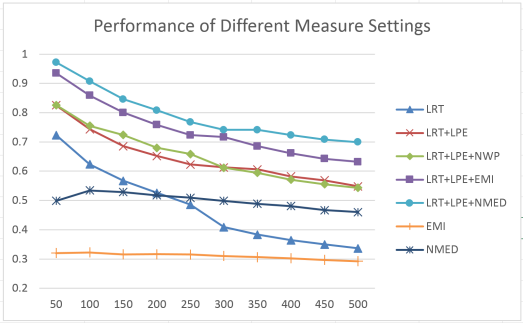


Figure 1: Comparative results of different measure settings. X-axis is the number of words returned (K), and Y-axis is average precision ($AP(K)$).

[3] All the resources are available upon request.

$top\ K\ words \Rightarrow$	100	200	300	400	500
LPE	0.366	0.324	0.286	0.270	0.259
LRT+LPE	**0.743**	**0.652**	**0.613**	**0.582**	**0.548**
LPE+NWP	0.467	0.400	0.350	0.330	0.320
LRT+LPE+NWP	**0.755**	**0.680**	**0.612**	**0.571**	**0.543**
LPE+EMI	0.608	0.551	0.519	0.486	0.467
LRT+LPE+EMI	**0.859**	**0.759**	**0.717**	**0.662**	**0.632**
LPE+NMED	0.749	0.690	0.641	0.612	0.576
LRT+LPE+NMED	**0.907**	**0.808**	**0.741**	**0.723**	**0.699**

Table 4: Results with vs. without likelihood ratio test (LRT).

4.4 Parameter Tuning

Firstly, we will show how to obtain the optimal settings of k_p and k_w. The measure setting we take here is $F_{NMED}(w)$, as shown in Formula (12). Again, we choose only one seed word "坑爹(reverse one's expectation)", and the number of words returned is set to $K = 300$. Results in Table 5 show that the performance drops consistently across different k_w settings when the number of patterns increases. Note that at the early stage of Algorithm 1, larger k_p (perhaps with noisy patterns) may lead to lower quality of new words; while larger k_w (perhaps with noisy seed words) may lead to lower quality of lexical patterns. Therefore, we choose the optimal setting to small numbers, as $k_p = 5, k_w = 10$.

Secondly, we justify whether the proposed algorithm is sensitive to the number of seed words. We set $k_p = 5$ and $k_w = 10$, and take F_{NMED} as the weighting measure of new word. We experimented with only one seed word, two, three, and four seed words, respectively. The results in Table 6 show very stable performance when different numbers of seed words are chosen. It's interesting that the performance is totally the same with different numbers of seed words. By looking into the pattern set and the selected words at each iteration, we found that the pattern set ($\mathcal{P}$) converges soon to the same set after a few iterations; and at the beginning several iterations, the selected words are almost the same although the order of adding the words is different. Since the algorithm will finally sort the words at step (11) and $\mathcal{P}$ is the same, the ranking of the words becomes all the same.

Lastly, we need to decide the optimal number of patterns in $\mathcal{P}_c$ (that is, k_c in Algorithm 1) because the set has been used in computing left pattern entropy, see Formula (4). Too small size of $\mathcal{P}_c$ may lead to insufficient estimation of left pattern entropy. Results in Table 7 shows that larger $\mathcal{P}_c$ decrease the performance, particularly when the number of words returned (K) becomes larger. Therefore, we set $|\mathcal{P}_c| = 100$.

4.5 Polarity Prediction of New Sentiment Words

In this section, we attempt to classifying the polarity of the annotated 323 new words. Two methods are adapted with different settings for this purpose. The first one is majority vote (MV), and the second one is pointwise mutual information, similar to (Turney and Littman, 2003). The majority vote method is formulated as below:

$$MV(w) = \sum_{w_p \in PW} \frac{\#(w, w_p)}{|PW|} - \sum_{w_n \in NW} \frac{\#(w, w_n)}{|NW|}$$

where PW and NW are a positive and negative set of emoticons (or seed words) respectively, and $\#(w, w_p)$ is the co-occurrence count of the input word w and the item w_p. The polarity is judged according to this rule: if $MV(w) > th_1$, the word w is positive; if $MV(w) < -th_1$ the word negative; otherwise neutral. The threshold th_1 is manually tuned.

And PMI is computed as follows:

$$PMI(w) = \sum_{w_p \in PW} \frac{PMI(w, w_p)}{|PW|} - \sum_{w_n \in NW} \frac{PMI(w, w_n)}{|NW|}$$

where $PMI(x, y) = log_2(\frac{Pr(x,y)}{Pr(x)*Pr(y)})$, and $Pr(\cdot)$ denotes probability. The polarity is judged according to the rule: if $PMI(w) > th_2$, w is positive; if $PMI(w) < -th_2$ negative; otherwise neutral. The threshold th_2 is manually tuned.

As for the resources PW and NW, we have three settings. The first setting (denoted by

k_w \ k_p	2	3	4	5	10	20	50
5	0.753	0.738	0.746	0.741	0.741	0.734	0.715
10	0.753	0.738	0.746	0.741	0.741	0.728	0.712
15	0.753	0.738	0.746	0.741	0.754	0.734	0.718
20	0.763	0.738	0.744	0.749	0.749	0.735	0.717

Table 5: Parameter tuning results for k_p and k_w. The measure setting is $F_{NMED}(w)$, the seed word set is {"坑爹(reverse one's expectation)"}, and the number of words returned is $K = 300$.

$\#\ seeds \Rightarrow$	1	2	3	4
K=100	0.907	0.907	0.907	0.907
K=200	0.808	0.808	0.808	0.808
K=300	0.741	0.741	0.741	0.741
K=400	0.709	0.709	0.709	0.709
K=500	0.685	0.685	0.685	0.685

Table 6: Performance with different numbers of seed words. The measure setting is $F_{NMED}(w)$, and $k_p = 5$, $k_w = 10$. The seed words are chosen from Table 1.

Large_Emo) is a set of most frequent 36 emoticons in which there are 21 positive and 15 negative emoticons respectively. The second one (denoted by Small_Emo) is a set of 10 emoticons, which are chosen from the 36 emoticons, as shown in Table 8. The third one (denoted by Opin_Words) is two sets of seed opinion words, where PW={高兴(happy),大方(generous),漂亮(beautiful), 善良(kind), 聪明(smart)} and NW ={伤心(sad),小气(mean),难看(ugly), 邪恶(wicked), 笨(stupid)}.

The performance of polarity prediction is shown in Table 9. In two-class polarity classification, we remove neutral words and only make prediction with positive/negative classes. The first observation is that the performance of using emoticons is much better than that of using seed opinion words. We conjecture that this may be because new sentiment words are more frequently co-occurring with emoticons than with these opinion words. The second observation is that three-class polarity classification is much more difficult than two-class polarity classification because many extracted new words are nouns such as "基友(gay)","菇凉(girl)", and "盆友(friend)". Such nouns are more difficult to classify sentiment orientation.

4.6 Application of New Sentiment Words to Sentiment Classification

In this section, we justify whether inclusion of new sentiment word would benefit sentiment classification. For this purpose, we randomly sampled and annotated 4,500 Weibo posts that contain at least one opinion word in the union of the Hownet [4] opinion lexicons and our annotated new words. We apply two models for polarity classification. The first model is a lexicon-based model (denoted by $Lexicon$) that counts the number of positive and negative opinion words in a post respectively, and classifies a post to be positive if there are more positive words than negative ones, and to be negative otherwise. The second model is a SVM model in which opinion words are used as feature, and 5-fold cross validation is conducted.

We experiment with different settings of Hownet lexicon resources:

- Hownet opinion words (denoted by Hownet): After removing some obviously inappropriate words, the left lexicons have 627 positive opinion words and 1,038 negative opinion words, respectively.
- Compact Hownet opinion words (denoted by cptHownet): we count the frequency of the above opinion words on the training data and remove words whose document frequency is less than 2. This results in 138 positive words and 125 negative words.

Then, we add into the above resources the labeled new polar words(denoted by NW, including 116 positive and 112 negative words) and the top 100 words produced by the algorithm (denoted by $T100$), respectively. Note that the lexicon-based model requires the sentiment orientation of each dictionary entry [5], we thus manually label the po-

[4] http://www.keenage.com/html/c_index.html.

[5] This is not necessary for the SVM model. All words in the top 100 words can be used as feature.

$\|\mathcal{P}_c\| \Rightarrow$	50	100	200	300	400	500
K=100	0.907	0.905	0.916	0.916	0.888	0.887
K=200	0.808	0.810	0.778	0.776	0.766	0.764
K=300	0.741	0.731	0.722	0.726	0.712	0.713
K=400	0.709	0.708	0.677	0.675	0.656	0.655
K=500	0.685	0.683	0.653	0.646	0.626	0.627

Table 7: Tuning the number of patterns in $\mathcal{P}_c$. The measure setting is $F_{NMED}(w)$, $k_p = 5$, $k_w = 10$, and the seed word set is {"坑爹(reverse one's expectation)"}.

Emoticon	Polarity	Emoticon	Polarity
	positive		negative
	positive		negative
	positive		negative
	positive		negative
	positive		negative

Table 8: The ten emoticons used for polarity prediction.

Methods $\Rightarrow$	Majority vote	PMI
Two-class polarity classification		
Large_Emo	0.861	0.865
Small_Emo	0.846	0.851
Opin_Words	0.697	0.654
Three-class polarity classification		
Large_Emo	0.598	0.632
Small_Emo	0.551	0.635
Opin_Words	0.449	0.486

Table 9: The accuracy of two/three-class polarity classification.

larity of all top 100 words (we did NOT remove incorrect new word). This results in 52 positive and 34 negative words.

Results in Table 10 show that inclusion of new words in both models improves the performance remarkably. In the setting of the original lexicon (Hownet), both models obtain 2-3% gains from the inclusion of new words. Similar improvement is observed in the setting of the compact lexicon. Note, that $T100$ is automatically obtained from Algorithm 1 so that it may contain words that are not new sentiment words, but the resource also improves performance remarkably.

5 Conclusion

In order to extract *new sentiment words* from large-scale user-generated content, this paper proposes a fully unsupervised, purely data-driven, and

	# Pos/Neg	Lexicon	SVM
Hownet	627/1,038	0.737	0.756
Hownet+NW	743/1,150	0.770	0.779
Hownet+T100	679/1,172	0.761	0.774
cptHownet	138/125	0.738	0.758
cptHownet+NW	254/237	0.774	0.782
cptHownet+T100	190/159	0.764	0.775

Table 10: The accuracy of polarity classfication of Weibo post with/without new sentiment words. N-W includes 116/112 positive/negative words, and T100 contains 52/34 positive/negative words.

almost knowledge-free (except POS tags) framework. We design statistical measures to quantify the utility of a lexical pattern and to measure the possibility of a word being a new word, respectively. The method is almost free of linguistic resources (except POS tags), and does not rely on elaborated linguistic rules. We conduct extensive experiments to reveal the influence of different statistical measures in new word finding. Comparative experiments show that our proposed method outperforms baselines remarkably. Experiments also demonstrate that inclusion of new sentiment words benefits sentiment classification definitely.

From linguistic perspectives, our framework is capable to extract *adjective* new words because the lexical patterns usually modify adjective words. As future work, we are considering how to extract other types of new sentiment words, such as *nounal* new words that can express sentiment.

Acknowledgments

This work was partly supported by the following grants from: the National Basic Research Program (973 Program) under grant No. 2012CB316301 and 2013CB329403, the National Science Foundation of China project under grant No. 61332007 and No. 60803075, and the Beijing Higher Education Young Elite Teacher Project.

References

Shlomo Argamon, Ido Dagan, and Yuval Krymolowski. 1998. A memory-based approach to learning shallow natural language patterns. In Proceedings of the 17th International Conference on Computational Linguistics - Volume 1, COLING '98, pages 67--73, Stroudsburg, PA, USA. Association for Computational Linguistics.

Fan Bu, Xiaoyan Zhu, and Ming Li. 2010. Measuring the non-compositionality of multiword expressions. In Proceedings of the 23rd International Conference on Computational Linguistics, COLING '10, pages 116--124, Stroudsburg, PA, USA. Association for Computational Linguistics.

Keh-Jiann Chen and Wei-Yun Ma. 2002. Unknown word extraction for chinese documents. In Proceedings of the 19th International Conference on Computational Linguistics - Volume 1, COLING '02, pages 1--7, Stroudsburg, PA, USA. Association for Computational Linguistics.

Aitao Chen. 2003. Chinese word segmentation using minimal linguistic knowledge. In Proceedings of the Second SIGHAN Workshop on Chinese Language Processing - Volume 17, SIGHAN '03, pages 148--151, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yaacov Choueka. 1988. Looking for needles in a haystack or locating interesting collocation expressions in large textual databases. In Proceeding of the RIAO'88 Conference on User-Oriented Content-Based Text and Image Handling, pages 21--24.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. Comput. Linguist., 16(1): 22--29, March.

J Ferreira da Silva and G Pereira Lopes. 1999. A local maxima method and a fair dispersion normalization for extracting multi-word units from corpora. In Sixth Meeting on Mathematics of Language, pages 369--381.

Gaël Dias, Sylvie Guilloré, and José Gabriel Pereira Lopes. 2000. Mining textual associations in text corpora. 6th ACM SIGKDD Work. Text Mining.

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. Comput. Linguist., 19(1):61--74, March.

Zhen Hai, Kuiyu Chang, and Gao Cong. 2012. One seed to find them all: Mining opinion features via association. In Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM '12, pages 255--264, New York, NY, USA. ACM.

John S Justeson and Slava M Katz. 1995. Technical terminology: some linguistic properties and an algorithm for identification in text. Natural language engineering, 1(1):9--27.

Hongqiao Li, Chang-Ning Huang, Jianfeng Gao, and Xiaozhong Fan. 2005. The use of svm for chinese new word identification. In Natural Language Processing--IJCNLP 2004, pages 723--732. Springer.

Pavel Pecina. 2005. An extensive empirical study of collocation extraction methods. In Proceedings of the ACL Student Research Workshop, ACLstudent '05, pages 13--18, Stroudsburg, PA, USA. Association for Computational Linguistics.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In Proceedings of the 20th International Conference on Computational Linguistics, COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2011. Opinion word expansion and target extraction through double propagation. Computational linguistics, 37(1):9--27.

Patrick Schone and Daniel Jurafsky. 2001. Is knowledge-free induction of multiword unit dictionary headwords a solved problem. In Proc. of the 6th Conference on Empirical Methods in Natural Language Processing (EMNLP 2001), pages 100--108.

Richard Sproat and Thomas Emerson. 2003. The first international chinese word segmentation bakeoff. In Proceedings of the Second SIGHAN Workshop on Chinese Language Processing - Volume 17, SIGHAN '03, pages 133--143, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xu Sun, Houfeng Wang, and Wenjie Li. 2012. Fast online training with frequency-adaptive learning rates for chinese word segmentation and new word detection. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1, ACL '12, pages 253--262, Stroudsburg, PA, USA. Association for Computational Linguistics.

Beijing Thesaurus Research Center. 2003. Xinhua Xin Ciyu Cidian. Commercial Press, Beijing.

Peter D. Turney and Michael L. Littman. 2003. Measuring praise and criticism: Inference of semantic orientation from association. ACM Trans. Inf. Syst., 21(4):315--346, October.

Hua-Ping Zhang, Hong-Kui Yu, De-Yi Xiong, and Qun Liu. 2003. Hhmm-based chinese lexical analyzer ictclas. In Proceedings of the Second SIGHAN Workshop on Chinese Language Processing -

Volume 17, SIGHAN '03, pages 184--187, Stroudsburg, PA, USA. Association for Computational Linguistics.

Wen Zhang, Taketoshi Yoshida, Xijin Tang, and Tu-Bao Ho. 2009. Improving effectiveness of mutual information for substantival multiword expression extraction. Expert Systems with Applications, 36(8):10919--10930.

Yan Zhang, Maosong Sun, and Yang Zhang. 2010. Chinese new word detection from query logs. In Advanced Data Mining and Applications, pages 233--243. Springer.

Yabin Zheng, Zhiyuan Liu, Maosong Sun, Liyun Ru, and Yang Zhang. 2009. Incorporating user behaviors in new word detection. In Proceedings of the 21st International Jont Conference on Artifical Intelligence, IJCAI'09, pages 2101--2106, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

GuoDong Zhou. 2005. A chunking strategy towards unknown word detection in chinese word segmentation. In Natural Language Processing--IJCNLP 2005, pages 530--541. Springer.

ReNew: A Semi-Supervised Framework for Generating Domain-Specific Lexicons and Sentiment Analysis

Zhe Zhang
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206
zzhang13@ncsu.edu

Munindar P. Singh
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206
singh@ncsu.edu

Abstract

The sentiment captured in opinionated text provides interesting and valuable information for social media services. However, due to the complexity and diversity of linguistic representations, it is challenging to build a framework that accurately extracts such sentiment. We propose a semi-supervised framework for generating a domain-specific sentiment lexicon and inferring sentiments at the *segment* level. Our framework can greatly reduce the human effort for building a domain-specific sentiment lexicon with high quality. Specifically, in our evaluation, working with just 20 manually labeled reviews, it generates a domain-specific sentiment lexicon that yields weighted average F-Measure gains of 3%. Our sentiment classification model achieves approximately 1% greater accuracy than a state-of-the-art approach based on elementary discourse units.

1 Introduction

Automatically extracting sentiments from user-generated opinionated text is important in building social media services. However, the complexity and diversity of the linguistic representations of sentiments make this problem challenging.

High-quality sentiment lexicons can improve the performance of sentiment analysis models over general-purpose lexicons (Choi and Cardie, 2009). More advanced methods such as (Kanayama and Nasukawa, 2006) adopt domain knowledge by extracting sentiment words from the domain-specific corpus. However, depending on the context, the same word can have different polarities even in the same domain (Liu, 2012).

In respect to sentiment classification, Pang et al. (2002) infer the sentiments using basic features, such as bag-of-words. To capture more complex linguistic phenomena, leading approaches (Nakagawa et al., 2010; Jo and Oh, 2011; Kim et al., 2013) apply more advanced models but assume one document or sentence holds one sentiment. However, this is often not the case. Sentiments can change within one document, one sentence, or even one clause. Also, existing approaches infer sentiments without considering the changes of sentiments within or between clauses. However, these changes can be successfully exploited for inferring fine-grained sentiments.

To address the above shortcomings of lexicon and granularity, we propose a semi-supervised framework named ReNew. (1) Instead of using sentences, ReNew uses *segments* as the basic units for sentiment classification. Segments can be shorter than sentences and therefore help capture fine-grained sentiments. (2) ReNew leverages the relationships between consecutive segments to infer their sentiments and automatically generates a domain-specific sentiment lexicon in a semi-supervised fashion. (3) To capture the contextual sentiment of words, ReNew uses dependency relation pairs as the basic elements in the generated sentiment lexicon.

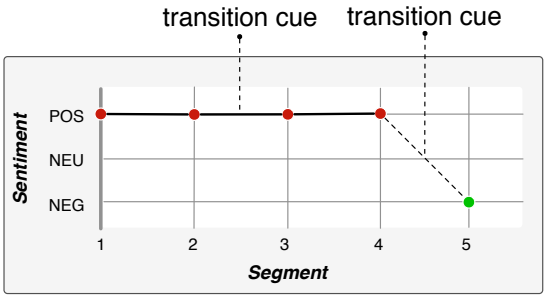


Figure 1: Segments in a Tripadvisor review.

Consider a part of a review from Tripadvisor.[1] We split it into six segments with sentiment labels.

[1] http://www.tripadvisor.com/ShowUserReviews-g32655-d81765-r100000013

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 542–551,
Baltimore, Maryland, USA, June 23-25 2014.

"... (**1: POS**) *The hotel was clean and comfortable.* (**2: POS**) *Service was friendly* (**3: POS**) *even providing us a late-morning check-in.* (**4: POS**) *The room was quiet and comfortable,* (**5: NEG**) *but it was beginning to show a few small signs of wear and tear. ...*"

Figure 1 visualizes the sentiment changes within the text. The sentiment remains the same across Segments 1 to 4. The sentiment transition between Segments 4 and 5 is indicated by the transition cue "but"—which signals conflict and contradiction. Assuming we know Segment 4 is positive, given the fact that Segment 5 starts with "but," we can infer with high confidence that the sentiment in Segment 5 changes to neutral or negative even without looking at its content. After classifying the sentiment of Segment 5 as NEG, we associate the dependency relation pairs {"sign", "wear"} and {"sign", "tear"} with that sentiment.

ReNew can greatly reduce the human effort for building a domain-specific sentiment lexicon with high quality. Specifically, in our evaluation on two real datasets, working with just 20 manually labeled reviews, ReNew generates a domain-specific sentiment lexicon that yields weighted average F-Measure gains of 3%. Additionally, our sentiment classification model achieves approximately 1% greater accuracy than a state-of-the-art approach based on elementary discourse units (Lazaridou et al., 2013).

The rest of this paper is structured as follows. Section 2 introduces some essential background. Section 3 illustrates ReNew. Section 4 presents our experiments and results. Section 5 reviews some related work. Section 6 concludes this paper and outlines some directions for future work.

2 Background

Let us introduce some of the key terminology used in ReNew. A **segment** is a sequence of words that represents at most one sentiment. A segment can consist of multiple consecutive clauses, up to a whole sentence. Or, it can be shorter than a clause. A **dependency relation** defines a binary relation that describes whether a pairwise syntactic relation among two words holds in a sentence. In ReNew, we exploit the Stanford typed dependency representations (de Marneffe et al., 2006) that use triples to formalize dependency relations. A domain-specific sentiment lexicon contains three lists of dependency relations, associated respectvely with positive, neutral, or negative sentiment.

Given a set of reviews, the tasks of sentiment analysis in ReNew are (1) splitting each review into segments, (2) associating each segment with a sentiment label (positive, neutral, negative), and (3) automatically generating a domain-specific sentiment lexicon. We employ Conditional Random Fields (Lafferty et al., 2001) to predict the sentiment label for each segment. Given a sequence of segments $\bar{x} = (x_1, \cdots, x_n)$ and a sequence of sentiment labels $\bar{y} = (y_1, \cdots, y_n)$, the CRFs model $p(\bar{y}|\bar{x})$ as follows.

$$p(\bar{y}|\bar{x}) = \frac{1}{Z(\bar{x})} \exp \sum_{j}^{J} (\omega_j \cdot F_j(\bar{x}, \bar{y}))$$

$$F_j(\bar{x}, \bar{y}) = \sum_{i=1}^{n} f_j(y_{i-1}, y_i, \bar{x}, i)$$

where ω is a set of weights learned in the training process to maximize $p(\bar{y}|\bar{x})$. $Z(\bar{x})$ is a normalization constant that is the sum of all possible label sequences. And, F_j is a feature function that sums f_j over $i \in (1, n)$, where n is the length of $\bar{y}$, and f_j can have arbitrary dependencies on the observation sequence $\bar{x}$ and neighboring labels.

3 Framework

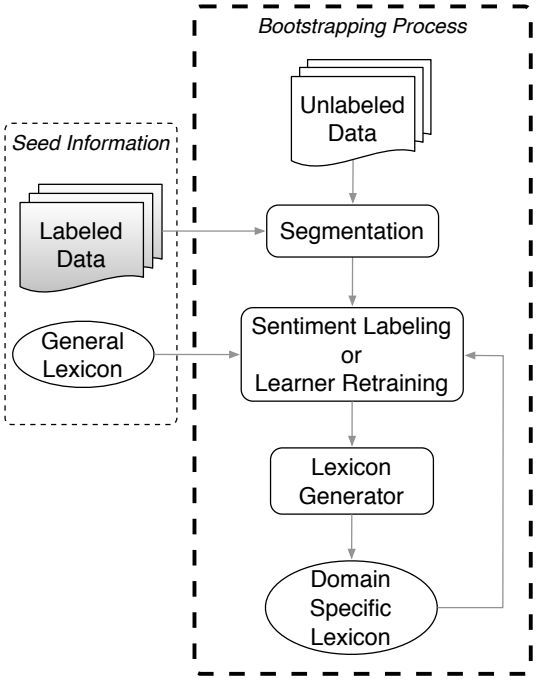


Figure 2: The ReNew framework schematically.

Figure 2 illustrates ReNew. Its inputs include

a general sentiment lexicon and a small labeled training dataset. We use a general sentiment lexicon and the training dataset as prior knowledge to build the initial learners.

On each iteration in the bootstrapping process, additional unlabeled data is first segmented. Second, the learners predict labels for segments based on current knowledge. Third, the lexicon generator determines which newly learned dependency relation triples to promote to the lexicon. At the end of each iteration, the learners are retrained via the updated lexicon so as to classify better on the next iteration. After labeling all of the data, we obtain the final version of our learners along with a domain-specific lexicon.

3.1 Rule-Based Segmentation Algorithm

Algorithm 1 Rule-based segmentation.

Require: Review dataset T
1: **for all** review r in T **do**
2: Remove HTML tags
3: Expand typical abbreviations
4: Mark special name-entities
5: **for all** sentence m in r **do**
6: **while** m contains a transition cue **and** m is not empty **do**
7: Extract subclause p that contains the transition cue
8: Add p as segment s into segment list
9: Remove p from m
10: **end while**
11: Add the remaining part in m as segment s into segment list
12: **end for**
13: **end for**

The algorithm starts with a review dataset T. Each review r from dataset T is first normalized by a set of hard-coded rules (lines 2–4) to remove unnecessary punctuations and HTML tags, expand typical abbreviations, and mark special name entities (e.g., replace a URL by #LINK# and replace a monetary amount "$78.99" by #MONEY#).

After the normalization step, it splits each review r into sentences, and each sentence into subclauses (lines 6–10) provided transition cues occur. In effect, the algorithm converts each review into a set of segments.

Note that ReNew captures and uses the sentiment changes. Therefore, our segmentation algorithm considers only two specific types of transition cues including contradiction and emphasis.

3.2 Sentiment Labeling

ReNew starts with a small labeled training set. Knowledge from this initial training set is not sufficient to build an accurate sentiment classification model or to generate a domain-specific sentiment lexicon. Unlabeled data contains rich knowledge, and it can be easily obtained. To exploit this resource, on each iteration, the sentiment labeling component, as shown in Figure 3, labels the data by using multiple learners and a label integrator. We have developed a forward (FR) and a backward relationship (BR) learner to learn relationships among segments.

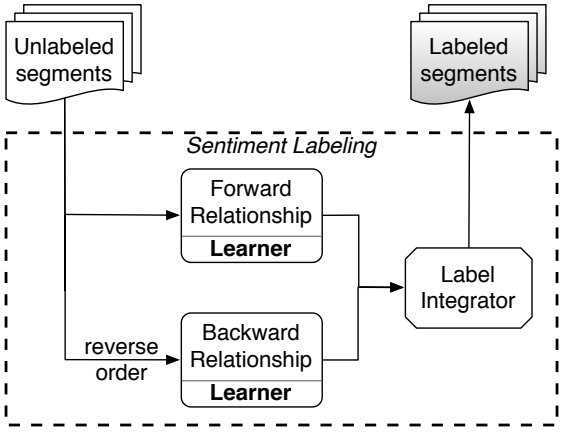


Figure 3: Sentiment labeling.

3.2.1 FR and BR Learners

The FR learner learns the relationship between the current segment and the next. Given the sentiment label and content of a segment, it tries to find the best possible sentiment label of the next segment. The FR Learner tackles the following situation where two segments are connected by a transition word, but existing knowledge is insufficient to infer the sentiment of the second segment. For instance, consider the following review sentence.[2]

(1) *The location is great,* (2) ***but*** *the staff was pretty ho-hum about everything from checking in, to AM hot coffee, to PM bar.*

The sentence contains two segments. We can easily infer the sentiment polarity of Segment 1 based on the word "great" that is commonly included in many general sentiment lexicons. For Segment 2, without any context information, it is difficult to infer its sentiment. Although the

[2]http://www.tripadvisor.com/ShowUserReviews-g60763-d93589-r10006597

word "ho-hum" indicates a negative polarity, it is not a frequent word. However, the conjunction "but" clearly signals a contrast. So, given the fact that the former segment is positive, a pre-trained FR learner can classify the latter as negative. The Backward Relationship (BR) learner does the same but with the segments in each review in reverse order.

3.2.2 Label Integrator

Given the candidate sentiment labels suggested by the two learners, the label integrator first selects the label with confidence greater than or equal to a preset threshold. Segments are left unlabeled if their candidate labels belong to mutually exclusive categories with the same confidence.

3.3 Lexicon Generator

In each iteration, after labeling a segment, the lexicon generator identifies new triples automatically. As shown in Figure 4, this module contains two parts: a Triple Extractor and a Lexicon Integrator. For each sentiment, the Triple Extractor (TE) extracts candidate dependency relation triples using a novel rule-based approach. The Lexicon Integrator (LI) evaluates the proposed candidates and promotes the most supported candidates to the corresponding sentiment category in the domain-specific lexicon.

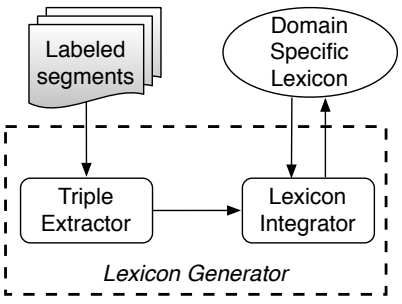


Figure 4: Lexicon generator module.

3.3.1 Triple Extractor (TE)

The TE follows the steps below, for segments that contain only one clause, as demonstrated in Figure 5 for *"The staff was slow and definitely not very friendly."* The extracted triples are $root\_nsubj$(slow, staff), $nsubj$(slow, staff), and $nsubj$(not_friendly, staff).

1. Generate a segment's dependency parse tree.
2. Identify the root node of each clause in the segment.
3. Remove all triples except those marked E in Table 1.
4. Apply the rules in Table 2 to add or modify triples.
5. Suggest the types of triples marked L in Table 1 to the lexicon integrator.

Table 1: Dependency relation types used in extracting (E) and domain-specific lexicon (L).

Types	Explanation	E	L
$amod$	adjectival modifier	√	√
$acomp$	adjectival complement	√	√
$nsubj$	nominal subject	√	√
neg	negation modifier	√	
$conj\_and$	words coordinated by "and" or similar	√	
$prep\_with$	words coordinated by "with"	√	
$root$	root node	√	
$root\_amod$	amod root node		√
$root\_acomp$	acomp root node		√
$root\_nsubj$	nsubj root node		√
$neg\_pattern$	"neg" pattern		√

Table 1 describes all seven types of triples used in the domain-specific lexicon. Among them, $amod$, $acomp$, and $nsubj$ are as in (de Marneffe et al., 2006). And, $root\_amod$ captures the root node of a sentence when it also appears in the adjectival modifier triple, similarly for $root\_acomp$ and $root\_nsubj$. We observe that the word of the root node is often related to the sentiment of a sentence and this is especially true when this word also appears in the adjectival modifier, adjectival complement, or negation modifier triple.

Zhang et al. (2010) propose the $no\_pattern$ that describes a word pair whose first word is "No" followed by a noun or noun phrase. They show that this pattern is a useful indicator for sentiment analysis. In our dataset, in addition to "No," we observe the frequent usage of "Nothing" followed by an adjective. For example, users may express a negative feeling about a hotel using sentence such as "Nothing special." Therefore, we create the $neg\_pattern$ to capture a larger range of possible word pairs. In ReNew, $neg\_pattern$ is "No" or "Nothing" followed by a noun or noun phrase or an adjective.

3.3.2 Lexicon Integrator (LI)

The Lexicon Integrator promotes candidate triples with a frequency greater than or equal to a preset

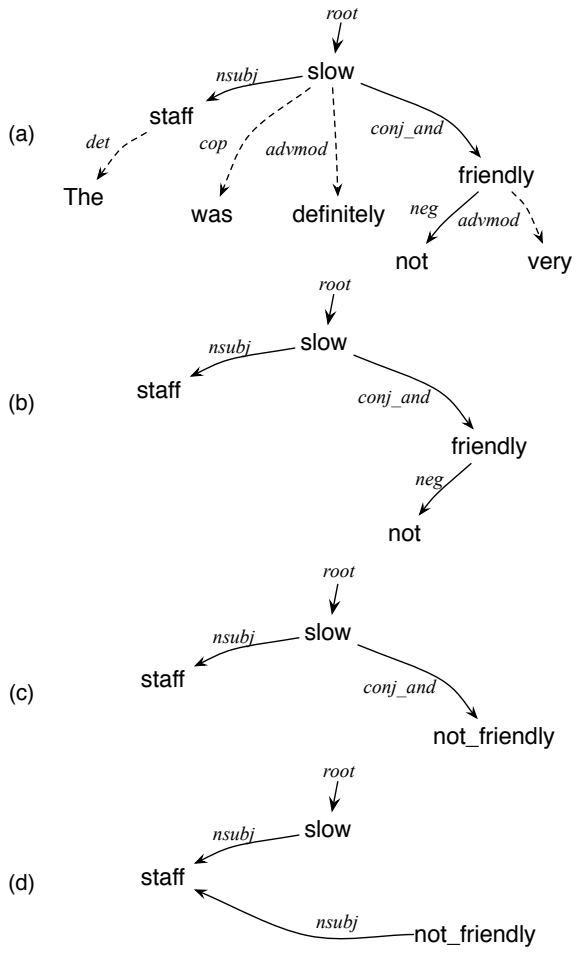


Figure 5: Extracting sentiment triples from a segment that contains one clause. (a) The initial dependency parse tree. (b) Remove nonsentiment triples. (c) Handle negation triples. (d) Build relationships.

threshold. The frequency list is updated in each iteration. The LI first examines the prior knowledge represented as an ordered list of the governors of all triples, each is attached with an ordered list of its dependents. Then, based on the triples promoted in this iteration, the order of the governors and their dependents is updated. Triples are not promoted if their governors or dependents appear in a predetermined list of stopwords.

The LI promotes triples by respecting mutual exclusion and the existing lexicon. In particular, it does not promote triples if they exist in multiple sentiment categories or if they already belong to a different sentiment category.

Finally, for each sentiment, we obtain seven sorted lists corresponding to *amod*, *acomp*, *nsubj*, *root_amod*, *root_acomp*, *root_nsubj*, and *neg_pattern*. These lists form the domain-specific sentiment lexicon.

Table 2: Rules for extracting sentiment triples.

Rule	Function	Condition	Result
R_1	Handle Negation	word w_i; $neg(w_{gov}, w_{dep})$; $w_i = w_{gov}$;	$w_i = w_{dep} + \text{“\_”} + w_i$
R_2	Build Relationships $(conj\_and, amod)$	word w_i and w_j; $conj\_and(w_i, w_j)$; $amod(w_{gov}, w_i)$;	$amod(w_{gov}, w_i)$ $amod(w_{gov}, w_j)$
R_3	Build Relationships $(conj\_and, acomp)$	word w_i and w_j; $conj\_and(w_i, w_j)$; $acomp(w_{gov}, w_i)$;	$acomp(w_{gov}, w_i)$ $acomp(w_{gov}, w_j)$
R_4	Build Relationships $(conj\_and, nsubj)$	word w_i and w_j; $conj\_and(w_i, w_j)$; $nsubj(w_i, w_{dep})$;	$nsubj(w_i, w_{dep})$ $nsubj(w_j, w_{dep})$

3.4 Learner Retraining

At the end of each iteration, ReNew retrains each learner as shown in Figure 6. Newly labeled segments are selected by a filter. Then, given an updated lexicon, learners are retrained to perform better on the next iteration. Detailed description of the filter and learner are presented below.

3.4.1 Filter

The filter seeks to prevent labeling errors from accumulating during bootstrapping. In ReNew, newly acquired training samples are segments with labels that are predicted by old learners. Each predicted label is associated with a confidence value. The filter is applied to select those labeled segments with confidence greater than or equal to a preset threshold.

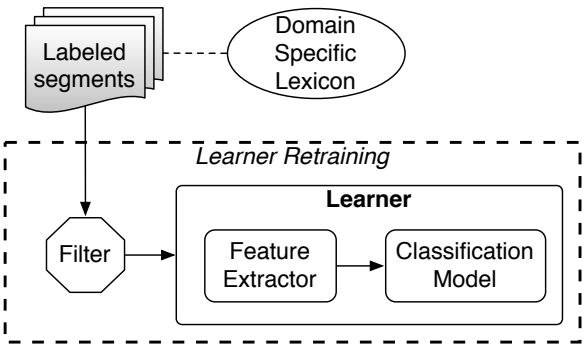


Figure 6: Retrain a relationship learner.

3.4.2 Learner

As Section 3.2 describes, ReNew uses learners to capture different types of relationships among segments to classify sentiment by leveraging these relationships. Each learner contains two components: a feature extractor and a classification model. To train a learner, the feature extractor first converts labeled segments into feature vectors

Table 3: A list of transition types used in ReNew.

Transition Types	Examples
Agreement, Addition, and Similarity	also, similarly, as well as, ...
Opposition, Limitation, and Contradiction	but, although, in contrast, ...
Cause, Condition, and Purpose	if, since, as/so long as, ...
Examples, Support, and Emphasis	including, especially, such as, ...
Effect, Consequence, and Result	therefore, thus, as a result ...
Conclusion, Summary, and Restatement	overall, all in all, to sum up, ...
Time, Chronology, and Sequence	until, eventually, as soon as, ...

for training a CRF-based sentiment classification model. The feature extractor generates five kinds of features as below.

Grammar: part-of-speech tag of every word, the type of phrases and clauses (if known).

Opinion word: To exploit a general sentiment lexicon, we use two binary features indicating the presence or absence of a word in the positive or negative list in a general sentiment lexicon.

Dependency relation: The lexicon generated by ReNew uses the Stanford typed dependency representation as its structure.

Transition cue: For tracking the changes of the sentiment, we exploit seven types of transition cues, as shown in Table 3.

Punctuation, special name-entity, and segment position: Some punctuation symbols, such as "!", are reliable carriers of sentiments. We mark special named-entities, such as time, money, and so on. In addition, we use segment positions (beginning, middle, and end) in reviews as features.

4 Experiments

To assess ReNew's effectiveness, we prepare two hotel review datasets crawled from Tripadvisor. One dataset contains a total of 4,017 unlabeled reviews regarding 802 hotels from seven US cities. The reviews are posted by 340 users, each of whom contributes at least ten reviews. The other dataset contains 200 reviews randomly selected from Tripadvisor. We collected ground-truth labels for this dataset by inviting six annotators in two groups of three. Each group labeled the same 100 reviews. We obtained the labels for each segment consist as positive, neutral, or negative. Fleiss' kappa scores for the two groups were 0.70 and 0.68, respectively, indicating substantial agreement between our annotators.

The results we present in the remainder of this section rely upon the following parameter values. The confidence thresholds used in the Label Integrator and filter are both set to 0.9 for positive labels and 0.7 for negative and neutral labels. The minimum frequency used in the Lexicon Integrator for selecting triples is set to 4.

4.1 Feature Function Evaluation

Our first experiment evaluates the effects of different combinations of features. To do this, we first divide all features into four basic feature sets: *T* (transition cues), *P* (punctuations, special name-entities, and segment positions), *G* (grammar), and *OD* (opinion words and dependency relations). We train 15 sentiment classification models using all basic features and their combinations. Figure 7 shows the results of a 10-fold cross validation on the 200-review dataset (light grey bars show the accuracy of the model trained without using transition cue features).

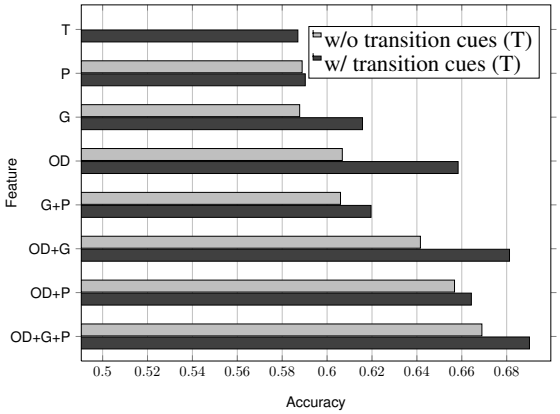


Figure 7: Accuracy using different features.

The feature *OD* yields the best accuracy, followed by *G*, *P*, and *T*. Although *T* yields the worst accuracy, incorporating it improves the resulting accuracy of the other features, as shown by the dark grey bars. In particular, the accuracy of *OD* is markedly improved by adding *T*. The model trained using all the feature sets yields the best accuracy.

4.2 Relationship Learners Evaluation

Our second experiment evaluates the impact of the relationship learners and the label integrator. To this end, we train and compare sentiment classification models using three configurations. The first configuration (FW-L) uses only the FR learner; the second (BW-L) only the BR learner. ALL-L uses both the FR and BR learners, together with a label integrator. We evaluate them with 10-fold cross

validation on the 200-review dataset.

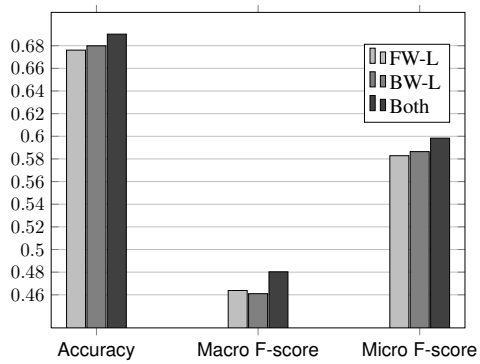


Figure 8: Comparison among the learners.

Figure 8 reports the accuracy, macro F-score, and micro F-score. It shows that the BR learner produces better accuracy and a micro F-score than the FR learner but a slightly worse macro F-score. Jointly considering both learners with the label integrator achieves better results than either alone. The results demonstrate the effectiveness of our sentiment labeling component.

4.3 Domain-Specific Lexicon Assessment

Our third experiment evaluates the quality of the domain-specific lexicon automatically generated by ReNew. To do this, we first transform each of the 200 labeled reviews into feature vectors. Then we retrain Logistic Regression models using WEKA (Hall et al., 2009). Note that we use only the features extracted from the lexicons themselves. This is important because to compare only the lexicons' impact on sentiment classification, we need to avoid the effect of other factors, such as syntax, transition cues, and so on. We compare models trained using (1) our domain-specific lexicon, (2) Affective Norms for English Words (ANEW) (Bradley and Lang, 1999), and (3) Linguistic Inquiry and Word Count (LIWC) (Tausczik and Pennebaker, 2010). ANEW and LIWC are well-known general sentiment lexicons.

Table 4 shows the results obtained by 10-fold cross validation. Each weighted average is computed according to the number of segments in each class. The table shows the significant advantages of the lexicon generated by ReNew. ANEW achieves the highest recall for the positive class, but the lowest recalls in the negative and neutral classes. Regarding the neutral class, both ANEW and LIWC achieve poor results. The weighted average measures indicate our lexicon has the highest overall quality.

Our domain-specific lexicon contains distinguishable aspects associated with sentiment words. For example, the aspect "staff" is associated with positive words (e.g., "nice," "friendli," "help," "great," and so on) and negative words (e.g., "okai," "anxiou," "moodi," "effici," and so on). We notice that some positive words also occur on the negative side. This may be for two reasons. First, some sentences that contain positive words may convey a negative sentiment, such as "The staff should be more efficient." Second, the bootstrapping process in ReNew may introduce some wrong words by mistakenly labeling the sentiment of the segments. These challenges suggest useful directions for the future work.

4.4 Lexicon Generation and Sentiment Classification

Our fourth experiment evaluates the robustness of ReNew's lexicon generation process as well as the performance of the sentiment classification models using these lexicons. We first generate ten domain-specific lexicons by repeatedly following these steps: For the first iteration, (1) build a training dataset by randomly selecting 20 labeled reviews (about 220 segments) and (2) train the learners using the training dataset and LIWC. For each iteration thereafter, (1) label 400 reviews from the unlabeled dataset (4,071 reviews) and (2) update the lexicon and retrain the learners. After labeling all of the data, output a domain-specific lexicon.

To evaluate the benefit of using domain-specific sentiment lexicons, we train ten sentiment classification models using the ten lexicons and then compare them, pairwise, against models trained with the general sentiment lexicon LIWC. Each model consists of an FR learner, a BR learner, and a label integrator. Each pairwise comparison is evaluated on a testing dataset with 10-fold cross validation. Each testing dataset consists of 180 randomly selected reviews (about 1,800 segments). For each of the pairwise comparisons, we conduct a paired t-test to determine if the domain-specific sentiment lexicon can yield better results.

Figure 9 shows the pairwise comparisons of accuracy between the two lexicons. Each group of bars represents the accuracy of two sentiment classification models trained using LIWC (CRFs-General) and the generated domain-specific lexicon (CRFs-Domain), respectively. The solid line corresponds to a baseline model that takes the ma-

Table 4: Comparison results of different lexicons.

	ANEW			LIWC			ReNew		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure
Positive	0.59	**0.994**	0.741	0.606	0.975	0.747	**0.623**	0.947	**0.752**
Negative	0.294	0.011	0.021	**0.584**	0.145	0.232	0.497	**0.202**	**0.288**
Neutral	0	0	0	0	0	0	**0.395**	**0.04**	**0.073**
Weighted average	0.41	0.587	0.44	0.481	0.605	0.489	**0.551**	**0.608**	**0.518**

jority classification strategy. Based on the distribution of the datasets, the majority class of all datasets is positive. We can see that models using either the general lexicon or the domain-specific lexicon achieve higher accuracy than the baseline model. Domain-specific lexicons produce significantly higher accuracy than general lexicons. In the figures below, we indicate significance to 10%, 5%, and 1% as '$\cdot$', '$*$', and '$**$', respectively.

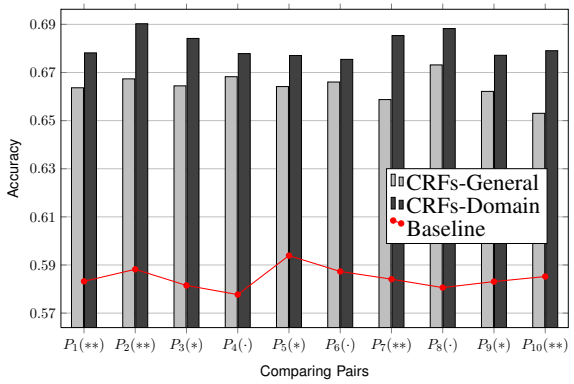


Figure 9: Accuracy with different lexicons.

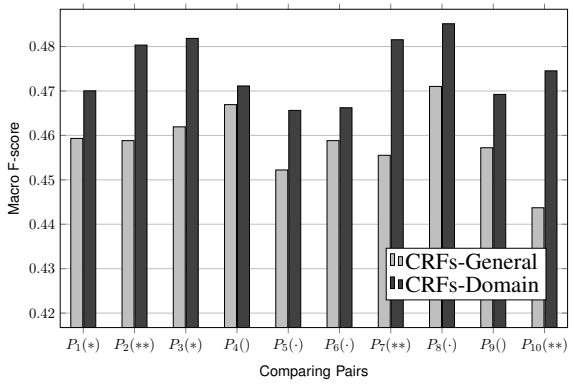


Figure 10: Macro F-score with different lexicons.

Figure 10 and 11 show the pairwise comparisons of macro and micro F-score together with the results of the paired t-tests. We can see that the domain-specific lexicons (dark-grey bars) consistently yield better results than their corresponding general lexicons (light-grey bars).

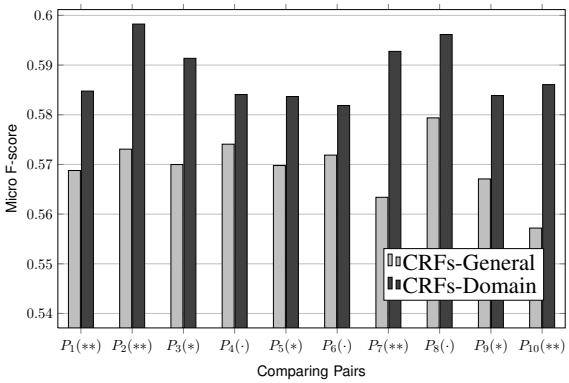


Figure 11: Micro F-score with different lexicons.

ReNew starts with LIWC and a labeled dataset and generates ten lexicons and sentiment classification models by iteratively learning 4,017 unlabeled reviews without any human guidance. The above results show that the generated lexicons contain more domain-related information than the general sentiment lexicons. Also, note that the labeled datasets we used contain only 20 labeled reviews. This is an easy requirement to meet.

4.5 Comparison with Previous Work

Our fifth experiment compares ReNew with Lazaridou et al.'s (2013) approach for sentiment classification using discourse relations. Like ReNew, Lazaridou et al.'s approach works on the sub sentential level. However, it differs from ReNew in three aspects. First, the basic units of their model are elementary discourse units (EDUs) from Rhetorical Structure Theory (RST) (Mann and Thompson, 1988). Second, their model considers the forward relationship between EDUs, whereas ReNew captures both forward and backward relationship between segments. Third, they use a generative model to capture the transition distributions over EDUs whereas ReNew uses a discriminative model to capture the transition sequences of segments.

EDUs are defined as minimal units of text and consider many more relations than the two types

Table 5: Comparison of our framework with previous work on sentiment classification.

Method	Accuracy
EDU-Model (Lazaridou et al.)	0.594
ReNew (our method)	**0.605**

of transition cues underlying our segments. We posit that EDUs are too fine-grained for sentiment analysis. Consider the following sentence from Lazaridou et al.'s dataset with its EDUs identified.

(1) *My husband called the front desk* (2) *to complain.*

Unfortunately, EDU (1) lacks sentiment and EDU (2) lacks the topic. Although Lazaridou et al.'s model can capture the forward relationship between any two consecutive EDUs, it cannot handle such cases because their model assumes that each EDU is associated with a topic and a sentiment. In contrast, ReNew finds just one segment in the above sentence.

Just to compare with Lazaridou et al., we apply our sentiment labeling component at the level of EDUs. Their labeled dataset contains 65 reviews, corresponding to 1,541 EDUs. Since this dataset is also extracted from Tripadvisor, we use the domain-specific lexicon automatically learned by ReNew based on our 4,071 unlabeled reviews. Follow the same training and testing regimen (10-fold cross validation), we compare ReNew with their model. As shown in Table 5, ReNew outperforms their approach on their dataset: Although ReNew is not optimized for EDUs, it achieves better accuracy.

5 Related Work

Two bodies of work are relevant. First, to generate sentiment lexicons, existing approaches commonly generate a sentiment lexicon by extending dictionaries or sentiment lexicons. Hu and Liu (2004), manually collect a small set of sentiment words and expand it iteratively by searching synonyms and antonyms in WordNet (Miller, 1995). Rao and Ravichandran (2009) formalize the problem of sentiment detection as a semi-supervised label propagation problem in a graph. Each node represents a word, and a weighted edge between any two nodes indicates the strength of the relationship between them. Esuli and Sebastiani (2006) use a set of classifiers in a semi-supervised fashion to iteratively expand a manually defined lexicon. Their lexicon, named SentiWordNet, comprises the synset of each word obtained from WordNet. Each synset is associated with three sentiment scores: positive, negative, and objective.

Second, for sentiment classification, Nakagawa et al. (2010) introduce a probabilistic model that uses the interactions between words within one sentence for inferring sentiments. Socher et al. (2011) introduce a semi-supervised approach that uses recursive autoencoders to learn the hierarchical structure and sentiment distribution of a sentence. Jo and Oh (2011) propose a probabilistic generative model named ASUM that can extract aspects coupled with sentiments. Kim et al. (2013) extend ASUM by enabling its probabilistic model to discover a hierarchical structure of the aspect-based sentiments. The above works apply sentence-level sentiment classification and their models are not able to capture the relationships between or among clauses.

6 Conclusions and Future Work

The leading lexical approaches to sentiment analysis from text are based on fixed lexicons that are painstakingly built by hand. There is little a priori justification that such lexicons would port across application domains. In contrast, ReNew seeks to automate the building of domain-specific lexicons beginning from a general sentiment lexicon and the iterative application of CRFs. Our results are promising. ReNew greatly reduces the human effort for generating high-quality sentiment lexicons together with a classification model. In future work, we plan to apply ReNew to additional sentiment analysis problems such as review quality analysis and sentiment summarization.

Acknowledgments

Thanks to Chung-Wei Hang, Chris Healey, James Lester, Steffen Heber, and the anonymous reviewers for helpful comments. This work is supported by the Army Research Laboratory in its Network Sciences Collaborative Technology Alliance (NS-CTA) under Cooperative Agreement Number W911NF-09-2-0053 and by an IBM Ph.D. Scholarship and an IBM Ph.D. fellowship.

References

Margaret M. Bradley and Peter J. Lang. 1999. Affective norms for English words (ANEW): Instruction manual and affective ratings. Technical Report C-1, The Center for Research in Psychophysiology, University of Florida, Gainesville, FL.

Yejin Choi and Claire Cardie. 2009. Adapting a polarity lexicon using integer linear programming for domain-specific sentiment classification. In *Proceedings of the* 14^{th} *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 590–598, Singapore.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the* 5^{th} *Conference on Language Resources and Evaluation (LREC)*, pages 449–454, Genoa, Italy.

Andrea Esuli and Fabrizio Sebastiani. 2006. SentiWordNet: A publicly available lexical resource for opinion mining. In *Proceedings of the* 5^{th} *Conference on Language Resources and Evaluation (LREC)*, pages 417–422, Genoa, Italy.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software: An update. *SIGKDD Explorations Newsletter*, 11(1):10–18, November.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the* 10^{th} *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 168–177, Seattle.

Yohan Jo and Alice Haeyun Oh. 2011. Aspect and sentiment unification model for online review analysis. In *Proceedings of the* 4^{th} *ACM International Conference on Web Search and Data Mining (WSDM)*, pages 815–824, Hong Kong.

Hiroshi Kanayama and Tetsuya Nasukawa. 2006. Fully automatic lexicon expansion for domain-oriented sentiment analysis. In *Proceedings of the* 11^{th} *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 355–363, Sydney.

Suin Kim, Jianwen Zhang, Zheng Chen, Alice H. Oh, and Shixia Liu. 2013. A hierarchical aspect-sentiment model for online reviews. In *Proceedings of the* 27^{th} *AAAI Conference on Artificial Intelligence (AAAI)*, pages 804–812, Bellevue.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the* 18^{th} *International Conference on Machine Learning (ICML)*, pages 282–289, San Francisco.

Angeliki Lazaridou, Ivan Titov, and Caroline Sporleder. 2013. A Bayesian model for joint unsupervised induction of sentiment, aspect and discourse representations. In *Proceedings of the* 51^{st} *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1630–1639, Sofia, Bulgaria.

Bing Liu. 2012. *Sentiment Analysis and Opinion Mining*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, San Rafael, CA.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

George A. Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.

Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. 2010. Dependency tree-based sentiment classification using CRFs with hidden variables. In *Proceedings of the* 11^{th} *Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 786–794, Los Angeles.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of the* 7^{th} *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 79–86, Philadelphia.

Delip Rao and Deepak Ravichandran. 2009. Semi-supervised polarity lexicon induction. In *Proceedings of the* 12^{th} *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 675–682, Athens.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the* 16^{th} *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 151–161, Edinburgh.

Yla R. Tausczik and James W. Pennebaker. 2010. The psychological meaning of words: LIWC and computerized text analysis methods. *Journal of Language and Social Psychology*, 29(1):24–54, March.

Lei Zhang, Bing Liu, Suk Hwan Lim, and Eamonn O'Brien-Strain. 2010. Extracting and ranking product features in opinion documents. In *Proceedings of the* 23^{rd} *International Conference on Computational Linguistics (COLING)*, pages 1462–1470, Beijing.

A Decision-Theoretic Approach to Natural Language Generation

Nathan McKinley
Department of EECS
Case Western Reserve University
Cleveland, OH, USA
nath@nmckinley.com

Soumya Ray
Department of EECS
Case Western Reserve University
Cleveland, OH, USA
sray@case.edu

Abstract

We study the problem of generating an English sentence given an underlying probabilistic grammar, a world and a communicative goal. We model the generation problem as a Markov decision process with a suitably defined reward function that reflects the communicative goal. We then use probabilistic planning to solve the MDP and generate a sentence that, with high probability, accomplishes the communicative goal. We show empirically that our approach can generate complex sentences with a speed that generally matches or surpasses the state of the art. Further, we show that our approach is anytime and can handle complex communicative goals, including negated goals.

1 Introduction

Suppose someone wants to tell their friend that they saw a dog chasing a cat. Given such a *communicative goal*, most people can formulate a sentence that satisfies the goal very quickly. Further, they can easily provide multiple similar sentences, differing in details but all satisfying the general communicative goal, with no or very little error. Natural language generation (NLG) develops techniques to extend similar capabilities to automated systems. In this paper, we study the restricted NLG problem: given a grammar, lexicon, world and a communicative goal, output a valid English sentence that satisfies this goal. The problem is restricted because in our work, we do not consider the issue of how to fragment a complex goal into multiple sentences (discourse planning).

Though restricted, this NLG problem is still difficult. A key source of difficulty is the nature of the grammar, which is generally large, probabilistic and ambiguous. Some NLG techniques use sampling strategies (Knight and Hatzivassiloglou, 1995) where a set of sentences is sampled from a data structure created from an underlying grammar and ranked according to how well they meet the communicative goal. Such approaches naturally handle statistical grammars, but do not solve the generation problem in a goal-directed manner. Other approaches view NLG as a *planning* problem (Koller and Stone, 2007). Here, the communicative goal is treated as a predicate to be satisfied, and the grammar and vocabulary are suitably encoded as logical operators. Then automated classical planning techniques are used to derive a plan which is converted into a sentence. This is an elegant formalization of NLG, however, restrictions on what current planning techniques can do limit its applicability. A key limitation is the logical nature of automated planning systems, which do not handle probabilistic grammars, or force ad-hoc approaches for doing so (Bauer and Koller, 2010). A second limitation comes from restrictions on the goal: it may be difficult to ensure that some specific piece of information should *not* be communicated, or to specify preferences over communicative goals, or specify general conditions, like that the sentence should be readable by a sixth grader. A third limitation comes from the search process: without strong heuristics, most planners get bogged down when given communicative goals that require chaining together long sequences of operators (Koller and Petrick, 2011).

In our work, we also view NLG as a planning problem. However, we differ in that our underlying formalism for NLG is a suitably defined *Markov decision process* (MDP). This setting allows us to address the limitations outlined

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 552–561,
Baltimore, Maryland, USA, June 23-25 2014.

above: it is naturally probabilistic, and handles probabilistic grammars; we are able to specify complex communicative goals and general criteria through a suitably-defined reward function; and, as we show in our experiments, recent developments in fast planning in large MDPs result in a generation system that can rapidly deal with very specific communicative goals. Further, our system has several other desirable properties: it is an anytime approach; with a probabilistic grammar, it can naturally be used to sample and generate multiple sentences satisfying the communicative goal; and it is robust to large grammar sizes. Finally, the decision-theoretic setting allows for a precise tradeoff between exploration of the grammar and vocabulary to find a better solution and exploitation of the current most promising (partial) solution, instead of a heuristic search through the solution space as performed by standard planning approaches.

Below, we first describe related work, followed by a detailed description of our approach. We then empirically evaluate our approach and a state-of-the-art baseline in several different experimental settings and demonstrate its effectiveness at solving a variety of NLG tasks. Finally, we discuss future extensions and conclude.

2 Related Work

Two broad lines of approaches have been used to attack the general NLG problem. One direction can be thought of as "overgeneration and ranking." Here some (possibly probabilistic) structure is used to generate multiple candidate sentences, which are then ranked according to how well they satisfy the generation criteria. This includes work based on chart generation and parsing (Shieber, 1988; Kay, 1996). These generators assign semantic meaning to each individual token, then use a set of rules to decide if two words can be combined. Any combination which contains a semantic representation equivalent to the input at the conclusion of the algorithm is a valid output from a chart generation system. Other examples of this idea are the HALogen/Nitrogen systems (Langkilde-Geary, 2002). HALogen uses a two-phase architecture where first, a "forest" data structure that compactly summarizes possible expressions is constructed. The structure allows for a more efficient and compact representation compared to lattice structures that were previously used in statistical sentence generation approaches. Using dynamic programming, the highest ranked sentence from this structure is then output. Many other systems using similar ideas exist, e.g. (White and Baldridge, 2003; Lu et al., 2009).

A second line of attack formalizes NLG as an AI planning problem. SPUD (Stone et al., 2003), a system for NLG through microplanning, considers NLG as a problem which requires realizing a deliberative process of goal-directed activity. Many such NLG-as-planning systems use a pipeline architecture, working from their communicative goal through discourse planning and sentence generation. In discourse planning, information to be conveyed is selected and split into sentence-sized chunks. These sentence-sized chunks are then sent to a *sentence generator*, which itself is usually split into two tasks, *sentence planning* and *surface realization* (Koller and Petrick, 2011). The sentence planner takes in a sentence-sized chunk of information to be conveyed and enriches it in some way. This is then used by a *surface realization* module which encodes the enriched semantic representation into natural language. This chain is sometimes referred to as the "NLG Pipeline" (Reiter and Dale, 2000).

Another approach, called *integrated generation*, considers both sentence generation portions of the pipeline together (Koller and Stone, 2007). This is the approach taken in some modern generators like CRISP (Koller and Stone, 2007) and PCRISP (Bauer and Koller, 2010). In these generators, the input semantic requirements and grammar are encoded in PDDL (Fox and Long, 2003), which an off-the-shelf planner such as Graphplan (Blum and Furst, 1997) uses to produce a list of applications of rules in the grammar. These generators generate parses for the sentence at the same time as the sentence, which keeps them from generating realizations that are grammatically incorrect, and keeps them from generating grammatical structures that cannot be realized properly.

In the NLG-as-planning framework, the choice of grammar representation is crucial in treating NLG as a planning problem; the grammar provides the actions that the planner will use to generate a sentence. Tree Adjoining Grammars (TAGs) are a common choice (Koller and Stone, 2007; Bauer and Koller, 2010). TAGs are tree-based grammars consisting of two sets of trees, called initial trees and auxiliary or adjoining trees. An

entire initial tree can replace a leaf node in the sentence tree whose label matches the label of the root of the initial tree in a process called "substitution." Auxiliary trees, on the other hand, encode recursive structures of language. Auxiliary trees have, at a minimum, a root node and a foot node whose labels match. The foot node must be a leaf of the auxiliary tree. These trees are used in a three-step process called "adjoining". The first step finds an adjoining location by searching through our sentence to find any subtree with a root whose label matches the root node of the auxiliary tree. In the second step, the target subtree is removed from the sentence tree, and placed in the auxiliary tree as a direct replacement for the foot node. Finally, the modified auxiliary tree is placed back in the sentence tree in the original target location. We use a variation of TAGs in our work, called a lexicalized TAG (LTAG), where each tree is associated with a lexical item called an anchor.

Though the NLG-as-planning approaches are elegant and appealing, a key drawback is the difficulty of handling probabilistic grammars, which are readily handled by the overgeneration and ranking strategies. Recent approaches such as PCRISP (Bauer and Koller, 2010) attempt to remedy this, but do so in a somewhat ad-hoc way, by transforming the probabilities into costs, because they rely on deterministic planning to actually realize the output. In this work, we directly address this by using a more expressive underlying formalism, a Markov decision process (MDP). We show empirically that this modification has other benefits as well, such as being anytime and an ability to handle complex communicative goals beyond those that deterministic planners can handle.

We note that prior work exists that uses MDPs for NLG (Lemon, 2011). That work differs from ours in several key respects: (i) it considers NLG at a coarse level, for example choosing the type of utterance (in a dialog context) and how to fill in specific slots in a template, (ii) the source of uncertainty is not language-related but comes from things like uncertainty in speech recognition, and (iii) the MDPs are solved using reinforcement learning and not planning, which is impractical in our setting. However, that work does consider NLG in the context of the broader task of dialog management, which we leave for future work.

3 Sentence Tree Realization with UCT

In this section, we describe our approach, called Sentence Tree Realization with UCT (STRUCT). We describe the inputs to STRUCT, followed by the underlying MDP formalism and the probabilistic planning algorithm we use to generate sentences in this MDP.

3.1 Inputs to STRUCT

STRUCT takes three inputs in order to generate a single sentence. These inputs are a grammar (including a lexicon), a communicative goal, and a world specification.

STRUCT uses a first-order logic-based semantic model in its communicative goal and world specification. This model describes named "entities," representing general things in the world. Entities with the same name are considered to be the same entity. These entities are described using first-order logic predicates, where the name of the predicate represents a statement of truth about the given entities. In this semantic model, the communicative goal is a list of these predicates with variables used for the entity names. For instance, a communicative goal of 'red(d), dog(d)' (in English, "say anything about a dog which is red.") would match a sentence with the semantic representation 'red(subj), dog(subj), cat(obj), chased(subj, obj)', like "The red dog chased the cat", for instance.

A grammar contains a set of PTAG trees, divided into two sets (initial and adjoining). These trees are annotated with the entities in them. Entities are defined as any element anchored by precisely one node in the tree which can appear in a statement representing the semantic content of the tree. In addition to this set of trees, the grammar contains a list of words which can be inserted into those trees, turning the PTAG into an PLTAG. We refer to this list as a lexicon. Each word in the lexicon is annotated with its first-order logic semantics with any number of entities present in its subtree as the arguments.

A world specification is simply a list of all statements which are true in the world surrounding our generation. Matching entity names refer to the same entity. We use the closed world assumption, that is, any statement not present in our world is false. Before execution begins, our grammar is pruned to remove entries which cannot possibly be used in generation for the given problem, by tran-

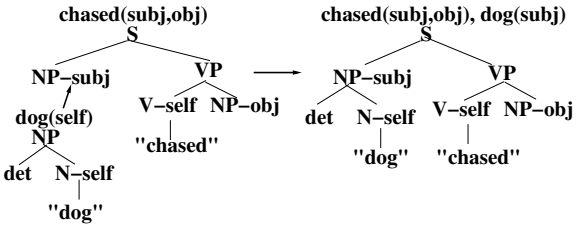


Figure 1: An example tree substitution operation in STRUCT.

sitively discovering all predicates that hold about the entities mentioned in the goal in the world, and eliminating all trees not about any of these. This often allows STRUCT to be resilient to large grammar sizes, as our experiments will show.

3.2 Specification of the MDP

We formulate NLG as a planning problem on a Markov decision process (MDP) (Puterman, 1994). An MDP is a tuple (S, A, T, R, γ) where S is a set of states, A is a set of actions available to an agent, $T : S \times A \times S \rightarrow [0, 1]$ is a possibly stochastic function defining the probability $T(s, a, s')$ with which the environment transitions to s' when the agent does a in state s. $R : S \times A \times S \rightarrow \mathbb{R}$ is a real-valued reward function that specifies the utility of performing action a in state s to reach another state. Finally, γ is a discount factor that allows planning over infinite horizons to converge. In such an MDP, the agent selects actions at each state to optimize the expected infinite-horizon discounted reward.

In the MDP we use for NLG, we must define each element of the tuple in such a way that a plan in the MDP becomes a sentence in a natural language. Our set of states, therefore, will be partial sentences which are in the language defined by our PLTAG input. There are an infinite number of these states, since TAG adjoins can be repeated indefinitely. Nonetheless, given a specific world and communicative goal, only a fraction of this MDP needs to be explored, and, as we show below, a good solution can often be found quickly using a variation of the UCT algorithm (Kocsis and Szepesvari, 2006).

Our set of actions consist of all single substitutions or adjoins at a particular valid location in the tree (example shown in Figure 1). Since we are using PLTAGs in this work, this means every action adds a word to the partial sentence. In situations where the sentence is complete (no nonterminals without children exist), we add a dummy action that the algorithm may choose to stop generation and emit the sentence. Based on these state and action definitions, the transition function takes a mapping between a partial sentence / action pair and the partial sentences which can result from one particular PLTAG adjoin / substitution, and returns the probability of that rule in the grammar.

In order to control the search space, we restrict the structure of the MDP so that while substitutions are available, only those operations are considered when determining the distribution over the next state, without any adjoins. We do this is in order to generate a complete and valid sentence quickly. This allows STRUCT to operate as an anytime algorithm, described further below.

The immediate value of a state, intuitively, describes closeness of an arbitrary partial sentence to our communicative goal. Each partial sentence is annotated with its semantic information, built up using the semantic annotations associated with the PLTAG trees. Thus we use as a reward a measure of the match between the semantic annotation of the partial tree and the communicative goal. That is, the larger the overlap between the predicates, the higher the reward. For an exact reward signal, when checking this overlap, we need to substitute each combination of entities in the goal into predicates in the sentence so we can return a high value if there are any mappings which are both possible (contain no statements which are not present in the grounded world) and mostly fulfill the goal (contain most of the goal predicates). However, this is combinatorial; also, most entities within sentences do not interact (e.g. if we say "the white rabbit jumped on the orange carrot," the whiteness of the rabbit has nothing to do with the carrot), and finally, an approximate reward signal generally works well enough unless we need to emit nested subclauses. Thus as an approximation, we use a reward signal where we simply count how many individual predicates overlap with the goal with *some* entity substitution. In the experiments, we illustrate the difference between the exact and approximate reward signals.

The final component of the MDP is the discount factor. We generally use a discount factor of 1; this is because we are willing to generate lengthy sentences in order to ensure we match our goal. A discount factor of 1 can be problematic in general since it can cause rewards to diverge, but since

there are a finite number of terms in our reward function (determined by the communicative goal and the fact that because of lexicalization we do not loop), this is not a problem for us.

3.3 The Probabilistic Planner

We now describe our approach to solving the MDP above to generate a sentence. Determining the optimal policy at *every* state in an MDP is polynomial in the size of the state-action space (Brafman and Tennenholtz, 2003), which is intractable in our case. But for our application, we do not need to find the optimal policy. Rather we just need to *plan* in an MDP to achieve a *given* communicative goal. Is it possible to do this without exploring the entire state-action space? Recent work answers this question affirmatively. New techniques such as sparse sampling (Kearns et al., 1999) and UCT (Kocsis and Szepesvari, 2006) show how to generate near-optimal plans in large MDPs with a time complexity that is independent of the state space size. Using the UCT approach with a suitably defined MDP (explained above) allows us to naturally handle probabilistic grammars as well as formulate NLG as a planning problem, unifying the distinct lines of attack described in Section 2. Further, the theoretical guarantees of UCT translate into fast generation in many cases, as we demonstrate in our experiments.

Online planning in MDPs as done by UCT follows two steps. From each state encountered, we construct a lookahead tree and use it to estimate the utility of each action in this state. Then, we take the best action, the system transitions to the next state and the procedure is repeated. In order to build a lookahead tree, we use a "rollout policy." This policy has two components: if it encounters a state already in the tree, it follows a "tree policy," discussed further below. If it encounters a new state, the policy reverts to a "default" policy that randomly samples an action. In all cases, any rewards received during the rollout search are backed up. Because this is a Monte Carlo estimate, typically, we run several simultaneous trials, and we keep track of the rewards received by each choice and use this to select the best action at the root.

The tree policy needed by UCT for a state s is the action a in that state which maximizes:

$$P(s,a) = Q(s,a) + c\sqrt{\frac{lnN(s)}{N(s,a)}} \qquad (1)$$

Algorithm 1 STRUCT algorithm.

Require: Number of simulations $numTrials$, Depth of lookahead $maxDepth$, time limit T
Ensure: Generated sentence tree

```
bestSentence ← nil
while time limit not reached do
    state ← empty sentence tree
    while state not terminal do
        for numTrials do
            testState ← state
            currentDepth ← 0
            if testState has unexplored actions then
                Apply one unexplored PLTAG production sampled from the PLTAG distribution to testState
                currentDepth++
            end if
            while currentDepth < maxDepth do
                Apply PLTAG production selected by tree policy (Equation 1) or default policy as required
                currentDepth++
            end while
            calculate reward for testState
            associate reward with first action taken
        end for
        state ← maximum reward testState
        if state score > bestSentence score and state has no nonterminal leaf nodes then
            bestSentence ← state
        end if
    end while
end while
return bestSentence
```

Here $Q(s,a)$ is the estimated value of a as observed in the tree search, computed as a sum over future rewards observed after (s,a). $N(s)$ and $N(s,a)$ are visit counts for the state and state-action pair. Thus the second term is an exploration term that biases the algorithm towards visiting actions that have not been explored enough. c is a constant that trades off exploration and exploitation. This essentially treats each action decision as a bandit problem; previous work shows that this approach can efficiently select near-optimal actions at each state.

We use a modified version of UCT in order to

increase its usability in the MDP we have defined. First, because we receive frequent, reasonably accurate feedback, we favor breadth over depth in the tree search. That is, it is more important in our case to try a variety of actions than to pursue a single action very deep. Second, UCT was originally used in an adversarial environment, and so is biased to select actions leading to the best average reward rather than the action leading to the best overall reward. This is not true for us, however, so we choose the latter action instead.

With the MDP definition above, we use our modified UCT to find a solution sentence (Algorithm 1). After every action is selected and applied, we check to see if we are in a state in which the algorithm could terminate (i.e. the sentence has no nonterminals yet to be expanded). If so, we determine if this is the best possibly-terminal state we have seen so far. If so, we store it, and continue the generation process. Whenever we reach a terminal state, we begin again from the start state of the MDP. Because of the structure restriction above (substitution before adjoin), STRUCT generates a valid sentence quickly. This enables STRUCT to perform as an anytime algorithm, which if interrupted will return the highest-value complete and valid sentence it has found. This also allows partial completion of communicative goals if not all goals can be achieved simultaneously in the time given.

4 Empirical Evaluation

In this section, we compare STRUCT to a state-of-the-art NLG system, CRISP, [1] and evaluate three hypotheses: (i) STRUCT is comparable in speed and generation quality to CRISP as it generates increasingly large referring expressions, (ii) STRUCT is comparable in speed and generation quality to CRISP as the size of the grammar which they use increases, and (iii) STRUCT is capable of communicating complex propositions, including multiple concurrent goals, negated goals, and nested subclauses.

For these experiments, STRUCT was implemented in Python 2.7. We used a 2010 version of CRISP which uses a Java-based GraphPlan implementation. All of our experiments were run on a 4-core AMD Phenom II X4 995 processor clocked at 3.2 GHz. Both systems were given access to 8 GB of RAM. The times reported are from the start of the generation process, eliminating variations due to interpreter startup, input parsing, etc.

[1] We were unfortunately unable to get the PCRISP system to compile, and so we could not evaluate it.

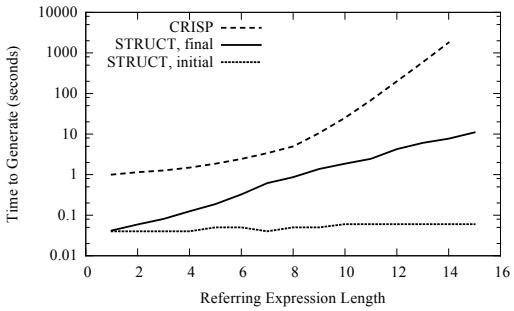


Figure 2: Experimental comparison between STRUCT and CRISP: Generation time vs. length of referring expression

4.1 Comparison to CRISP

We begin by describing experiments comparing STRUCT to CRISP. For these experiments, we use the approximate reward function for STRUCT.

Referring Expressions We first evaluate CRISP and STRUCT on their ability to generate referring expressions. Following prior work (Koller and Petrick, 2011), we consider a series of sentence generation problems which require the planner to generate a sentence like "The Adj_1 Adj_2 ... Adj_k dog chased the cat.", where the string of adjectives is a string that distinguishes one dog (whose identity is specified in the problem description) from all other entities in the world. In this experiment, $maxDepth$ was set equal to 1, since each action taken improved the sentence in a way measurable by our reward function. $numTrials$ was set equal to $k(k+1)$, since this is the number of adjoining sites available in the final step of generation, times the number of potential words to adjoin. This allows us to ensure successful generation in a single loop of the STRUCT algorithm.

The experiment has two parameters: j, the number of adjectives in the grammar, and k, the number of adjectives necessary to distinguish the entity in question from all other entities. We set $j = k$ and show the results in Figure 2. We observe that CRISP was able to achieve sub-second or similar times for all expressions of less than length 5, but its generation times increase exponentially past that point, exceeding 100 seconds for some plans at length 10. At length 15, CRISP failed to generate a referring expression;

after 90 minutes the Java garbage collector terminated the process. STRUCT (the "STRUCT_final" line) performs much better and is able to generate much longer referring expressions without failing. Later experiments had successful referring expression generation of lengths as high as 25. The "STRUCT_initial" curve shows the time taken by STRUCT to come up with the first complete sentence, which partially solves the goal and which (at least) could be output if generation was interrupted and no better alternative was found. As can be seen, this always happens very quickly.

Grammar Size. We next evaluate STRUCT and CRISP's ability to handle larger grammars. This experiment is set up in the same way as the one above, with the exception of l "distracting" words, words which are not useful in the sentence to be generated. l is defined as $j - k$. In these experiments, we vary l between 0 and 50. Figure 3a shows the results of these experiments. We observe that CRISP using GraphPlan, as previously reported in (Koller and Petrick, 2011), handles an increase in number of unused actions very well. Prior work reported a difference on the order of single milliseconds moving from $j = 1$ to $j = 10$. We report similar variations in CRISP runtime as j increases from 10 to 60: runtime increases by approximately 10% over that range.

No Pruning. If we do not prune the grammar (as described in Section 3.1), STRUCT's performance is similar to CRISP using the FF planner (Hoffmann and Nebel, 2001), also profiled in (Koller and Petrick, 2011), which increased from 27 ms to 4.4 seconds over the interval from $j = 1$ to $j = 10$. STRUCT's performance is less sensitive to larger grammars than this, but over the same interval where CRISP increases from 22 seconds of runtime to 27 seconds of runtime, STRUCT increases from 4 seconds to 32 seconds. This is due almost entirely to the required increase in the value of $numTrials$ as the grammar size increases. At the low end, we can use $numTrials = 20$, but at $l = 50$, we must use $numTrials = 160$ in order to ensure perfect generation as soon as possible. Note that, as STRUCT is an anytime algorithm, valid sentences are available very early in the generation process, despite the size of the set of adjoining trees. This time does not change substantially with increases in grammar size. However, the time to perfect this solution does.

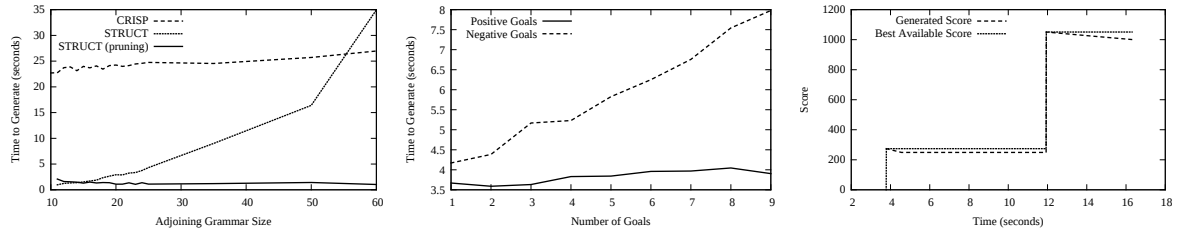
With Pruning. STRUCT's performance improves significantly if we allow for pruning. This experiment involving distracting words is an example of a case where pruning will perform well. When we apply pruning, we find that STRUCT is able to ignore the effect of additional distracting words. Experiments showed roughly constant times for generation for $j = 1$ through $j = 5000$. Our experiments do not show any significant impact on runtime due to the pruning procedure itself, even on large grammars.

4.2 Complex Communicative Goals

In the next set of experiments, we illustrate that STRUCT can solve a variety of complex communicative goals such as negated goals, conjuctions and goals requiring nested subclauses to be output.

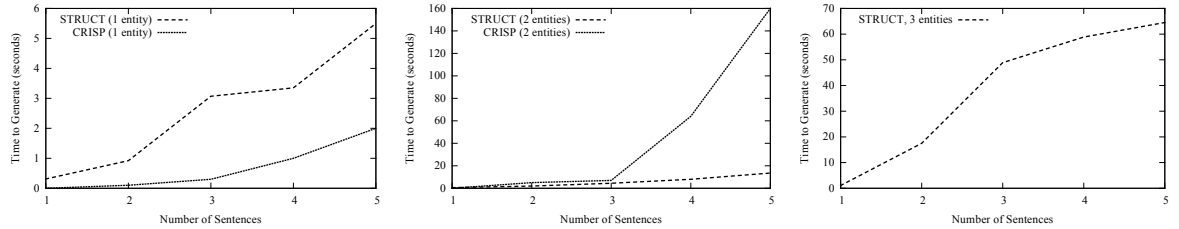
Multiple Goals. We first evaluate STRUCT's ability to accomplish multiple communicative goals when generating a single sentence. In this experiment, we modify the problem from the previous section. In that section, the referred-to dog was unique, and it was therefore possible to produce a referring expression which identified it unambiguously. In this experiment, we remove this condition by creating a situation in which the generator will be forced to ambiguously refer to several dogs. We then add to the world a number of adjectives which are common to each of these possible referents. Since these adjectives do not further disambiguate their subject, our generator should not use them in its output. We then encode these adjectives into communicative goals, so that they will be included in the output of the generator despite not assisting in the accomplishment of disambiguation. For example, assume we had two black cats, and we wanted to say that one of them was sleeping, but we wanted to emphasize that it was a black cat. We would have as our goal both "sleeps(c)" and "black(c)". We want the generator to say "the black cat sleeps", instead of simply "the cat sleeps."

We find that, in all cases, these otherwise useless adjectives are included in the output of our generator, indicating that STRUCT is successfully balancing multiple communicative goals. As we show in figure 3b (the "Positive Goals" curve) , the presence of additional satisfiable semantic goals does not substantially affect the time required for generation. We are able to accomplish this task with the same very high frequency as the CRISP



(a) Effect of grammar size (b) Effect of multiple/ negated goals (c) Effect of nested subclauses

Figure 3: STRUCT experiments (see text for details).



(a) One entity ("The man sat and the girl sat and ..."). (b) Two entities ("The dog chased the cat and ..."). (c) Three entities ("The man gave the girl the book and ...").

Figure 4: Time taken by STRUCT to generate sentences with conjunctions with varying numbers of entities.

comparisons, as we use the same parameters.

Negated Goals. We now evaluate STRUCT's ability to generate sentences given negated communicative goals. We again modify the problem used earlier by adding to our lexicon several new adjectives, each applicable only to the target of our referring expression. Since our target can now be referred to unambiguously using only one adjective, our generator should just select one of these new adjectives (we experimentally confirmed this). We then encode these adjectives into negated communicative goals, so that they will not be included in the output of the generator, despite allowing a much shorter referring expression. For example, assume we have a tall spotted black cat, a tall solid-colored white cat, and a short spotted brown cat, but we wanted to refer to the first one without using the word "black".

We find that these adjectives which should have been selected immediately are omitted from the output, and that the sentence generated is the best possible under the constraints. This demonstrates that STRUCT is balancing these negated communicative goals with its positive goals. Figure 3b (the "Negative Goals" curve) shows the impact of negated goals on the time to generation. Since this experiment alters the grammar size, we see the time to final generation growing linearly with grammar size. The increased time to generate can be traced directly to this increase in grammar size. This is a case where pruning does not help us in reducing the grammar size; we cannot optimistically prune out words that we do not plan to use. Doing so might reduce the ability of STRUCT to produce a sentence which partially fulfills its goals.

Nested subclauses. Next, we evaluate STRUCT's ability to generate sentences with nested subclauses. An example of such a sentence is "The dog which ate the treat chased the cat." This is a difficult sentence to generate for several reasons. The first, and clearest, is that there are words in the sentence which do not help to increase the score assigned to the partial sentence. Notably, we must adjoin the word "which" to "the dog" during the portion of generation where the sentence reads "the dog chased the cat". This decision requires us to do planning deeper than one level in the MDP, which increases the number of simulations STRUCT requires in order to get the correct result. In this case, we require lookahead further into the tree than depth 1. We need to know that using "which" will allow us to further specify which dog is chasing the cat; in order to do this we must use at least $d = 3$. Our reward function must determine this with, at a minimum, the actions corresponding to "which", "ate", and "treat". For these experiments, we use the exact reward function for STRUCT.

Despite this issue, STRUCT is capable of generating these sentences. Figure 3c shows the score of STRUCT's generated output over time for two nested clauses. Notice that, because the exact reward function is being used, the time to generate is longer in this experiment. To the best of our knowledge, CRISP is not able to generate sentences of this form due to an insufficiency in the way it handles TAGs, and consequently we present our results without this baseline.

Conjunctions. Finally, we evaluate STRUCT's ability to generate sentences including conjunctions. We introduce the conjunction "and", which allows for the root nonterminal of a new sentence ('S') to be adjoined to any other sentence. We then provide STRUCT with multiple goals. Given sufficient depth for the search ($d = 3$ was sufficient for our experiments, as our reward signal is fine-grained), STRUCT will produce two sentences joined by the conjunction "and". Again, we follow prior work in our experiment design (Koller and Petrick, 2011).

As we can see in Figures 4a, 4b, and 4c, STRUCT successfully generates results for conjunctions of up to five sentences. This is not a hard upper bound, but generation times begin to be impractically large at that point. Fortunately, human language tends toward shorter sentences than these unwieldy (but technically grammatical) sentences.

STRUCT increases in generation time both as the number of sentences increases and as the number of objects per sentences increases. We compare our results to those presented in (Koller and Petrick, 2011) for CRISP with the FF Planner. They attempted to generate sentences with three entities and failed to find a result within their 4 GB memory limit. As we can see, CRISP generates a result slightly faster than STRUCT when we are working with a single entity, but works much much slower for two entities and cannot generate results for a third entity. According to Koller's findings, this is because the search space grows by a factor of the universe size with the addition of another entity (Koller and Petrick, 2011).

5 Conclusion

We have proposed STRUCT, a general-purpose natural language generation system which is comparable to current state-of-the-art generators. STRUCT formalizes the generation problem as an MDP and applies a version of the UCT algorithm, a fast online MDP planner, to solve it. Thus, STRUCT naturally handles probabilistic grammars. We demonstrate empirically that STRUCT is anytime, comparable to existing generation-as-planning systems in certain NLG tasks, and is also capable of handling other, more complex tasks such as negated communicative goals.

Though STRUCT has many interesting properties, many directions for exploration remain. Among other things, it would be desirable to integrate STRUCT with discourse planning and dialog systems. Fortunately, reinforcement learning has already been investigated in such contexts (Lemon, 2011), indicating that an MDP-based generation procedure could be a natural fit in more complex generation systems. This is a primary direction for future work. A second direction is that, due to the nature of the approach, STRUCT is highly amenable to parallelization. None of the experiments reported here use parallelization, however, to be fair to CRISP. We plan to parallelize STRUCT in future work, to take advantage of current multicore architectures. This should obviously further reduce generation time.

STRUCT is open source and available from github.com upon request.

Acknowledgments

This work was supported in part by NSF CNS-1035602. SR was supported in part by CWRU award OSA110264. The authors are grateful to Umang Banugaria for help with the STRUCT implementation.

References

D. Bauer and A. Koller. 2010. Sentence generation as planning with probabilistic LTAG. *Proceedings of the 10th International Workshop on Tree Adjoining Grammar and Related Formalisms, New Haven, CT*.

A.L. Blum and M.L. Furst. 1997. Fast planning through planning graph analysis. *Artificial intelligence*, 90(1):281–300.

R. I. Brafman and M. Tennenholtz. 2003. R-MAX-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231.

M. Fox and D. Long. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20:61–124.

Jorg Hoffmann and Bernhard Nebel. 2001. The FF planning system: fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14(1):253–302, May.

Martin Kay. 1996. Chart generation. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, ACL '96, pages 200–204, Stroudsburg, PA, USA. Association for Computational Linguistics.

M. Kearns, Y. Mansour, and A.Y. Ng. 1999. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. In *International Joint Conference on Artificial Intelligence*, volume 16, pages 1324–1331. Lawrence Erlbaum Associates Ltd.

K. Knight and V. Hatzivassiloglou. 1995. Two-level, many-paths generation. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 252–260. Association for Computational Linguistics.

Levente Kocsis and Csaba Szepesvari. 2006. Bandit based Monte-Carlo planning. In *Proceedings of the Seventeenth European Conference on Machine Learning*, pages 282–293. Springer.

Alexander Koller and Ronald P. A. Petrick. 2011. Experiences with planning for natural language generation. *Computational Intelligence*, 27(1):23–40.

A. Koller and M. Stone. 2007. Sentence generation as a planning problem. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, volume 45, page 336.

I. Langkilde-Geary. 2002. An empirical verification of coverage and correctness for a general-purpose sentence generator. In *Proceedings of the 12th International Natural Language Generation Workshop*, pages 17–24.

Oliver Lemon. 2011. Learning what to say and how to say it: joint optimization of spoken dialogue management and natural language generation. *Computer Speech and Language*, 25(2):210–221.

W. Lu, H.T. Ng, and W.S. Lee. 2009. Natural language generation with tree conditional random fields. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 400–409. Association for Computational Linguistics.

M.L. Puterman. 1994. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, Inc.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press, January.

Stuart M. Shieber. 1988. A uniform architecture for parsing and generation. In *Proceedings of the 12th conference on Computational linguistics - Volume 2*, COLING '88, pages 614–619, Stroudsburg, PA, USA. Association for Computational Linguistics.

Matthew Stone, Christine Doran, Bonnie Webber, Tonia Bleam, and Martha Palmer. 2003. Microplanning with communicative intentions: The SPUD system. *Computational Intelligence*, 19(4):311–381.

M. White and J. Baldridge. 2003. Adapting chart realization to CCG. In *Proceedings of the 9th European Workshop on Natural Language Generation*, pages 119–126.

Generating Code-switched Text for Lexical Learning

Igor Labutov
Cornell University
iil4@cornell.edu

Hod Lipson
Cornell University
hod.lipson@cornell.edu

Abstract

A vast majority of L1 vocabulary acquisition occurs through incidental learning during reading (Nation, 2001; Schmitt et al., 2001). We propose a probabilistic approach to generating code-mixed text as an L2 technique for increasing retention in adult lexical learning through reading. Our model that takes as input a bilingual dictionary and an English text, and generates a code-switched text that optimizes a defined "learnability" metric by constructing a factor graph over lexical mentions. Using an artificial language vocabulary, we evaluate a set of algorithms for generating code-switched text automatically by presenting it to *Mechanical Turk* subjects and measuring recall in a sentence completion task.

1 Introduction

Today, an adult trying to learn a new language is likely to embrace an age-old and widely accepted practice of learning vocabulary through curated word lists and rote memorization. Yet, it is not uncommon to find yourself surrounded by speakers of a foreign language and instinctively pick up words and phrases without ever seeing the definition in your native tongue. Hearing "pass *le sale* please" at the dinner table from your in-laws visiting from abroad, is unlikely to make you think twice about passing the salt. Humans are extraordinarily good at inferring meaning from context, whether this context is your physical surrounding, or the surrounding text in the paragraph of the word that you don't yet understand.

Recently, a novel method of L2 language teaching had been shown effective in improving adult lexical acquisition rate and retention [1]. This technique relies on a phenomenon that elicits a natural simulation of L1-like vocabulary learning in adults — significantly closer to L1 learning for L2 learners than any model studied previously. By infusing foreign words into text in the learner's native tongue into low-surprisal contexts, the lexical acquisition process is facilitated naturally and non-obtrusively. Incidentally, this phenomenon occurs "in the wild" and is termed *code-switching* or *code-mixing*, and refers to the linguistic pattern of bilingual speakers swapping words and phrases between two languages during speech. While this phenomenon had received significant attention from both a socio-linguistic (Milroy and Muysken, 1995) and theoretical linguistic perspectives (Belazi et al., 1994; Bhatt, 1997) (including some computational studies), only recently has it been hypothesizes that "code-switching" is a marking of bilingual proficiency, rather than deficiency (Genesee, 2001).

Until recently it was widely believed that incidental lexical acquisition through reading can only occur for words that occur at sufficient density in a single text, so as to elicit the "noticing" effect needed for lexical acquisition to occur (Cobb, 2007). Recent neurophysiological findings, however, indicate that even a single incidental exposure to a novel word in a sufficiently constrained context is sufficient to trigger an early integration of the word in the brain's semantic network (Borovsky et al., 2012).

An approach explored in this paper, and motivated by the above findings, exploits "constraining" contexts in text to introduce novel words. A state-of-the-art approach for generating such text is based on an expert annotator whose job is to decide which words to "switch out" with novel foreign words (from hereon we will refer to the "switched out" word as the *source* word and to the "switched in" word as the *target* word). Consequently the process is labor-intensive and leads to

[1] authors' unpublished work

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 562–571,
Baltimore, Maryland, USA, June 23-25 2014.

a "one size fits all solution" that is insensitive to the learner's skill level or vocabulary proficiency. This limitation is also cited in literature as a significant roadblock to the widespread adaptation of graded reading series (Hill, 2008). A reading-based tool that follows the same principle, i.e. by systematic exposure of a learner to an incrementally more challenging text, will result in more effective learning (Lantolf and Appel, 1994).

To address the above limitation, we develop an approach for automatically generating such "code-switched" text with an explicit goal of maximizing the lexical acquisition rate in adults. Our method is based on a global optimization approach that incorporates a "knowledge model" of a user with the content of the text, to generate a sequence of lexical "switches". To facilitate the selection of "switch points", we learn a discriminative model for predicting switch point locations on a corpus that we collect for this purpose (and release to the community). Below is a high-level outline of this paper.

- We formalize our approach within a probabilistic graphical model framework, inference in which yields "code-switched" text that maximizes a surrogate to the acquisition rate objective.
- We compare this global method to several baseline techniques, including the strong "high-frequency" baseline.
- We analyze the operating range in which our model is effective and motivate the near-future extension of this approach with the proposed improvements.

2 Related Work

Our proposed approach to the computational generation of code-switched text, for the purpose of L2 pedagogy, is influenced by a number of fields that studied aspects of this phenomenon from distinct perspectives. In this section, we briefly describe a motivation from the areas of socio- and psycho- linguistics and language pedagogy research that indicate the promise of this approach.

2.1 Code-switching as a natural phenomenon

Code-switching (or code-mixing) is a widely studied phenomenon that received significant attention over the course of the last three decades, across the disciplines of sociolinguistics, theoretical and psycholinguistics and even literary and cultural studies (predominantly in the domain of *Spanish-English* code-switching) (Lipski, 2005).

Code-switching that occurs naturally in bilingual populations, and especially in children, has for a long time been considered a marking of incompetency in the second language. A more recent view on this phenomenon, however, suggests that due to the underlying syntactic complexity of code-switching, code-switching is actually a marking of bilingual fluency (Genesee, 2001). More recently, the idea of employing code-switching in the classroom, in a form of conversation-based exercises, has attracted the attention of multiple researchers and educators (Moodley, 2010; Macaro, 2005), yielding promising results in an elementary school study in South-Africa.

2.2 Computational Approaches to Code-switching

Additionally, there has been a limited number of studies of the computational approaches to code-switching, and in particular code-switched text generation. Solorio and Liu (2008), record and transcribe a corpus of Spanish-English code-mixed conversation to train a generative model (Naive Bayes) for the task of predicting code-switch points in conversation. Additionally they test their trained model in its ability to generate code-switched text with convincing results. Building on their work, (Adel et al., 2012) employ additional features and a recurrent network language model for modeling code-switching in conversational speech. Adel and collegues (2011) propose a statistical machine translation-based approach for generating code-switched text. We note, however, that the primary goal of these methods is in the faithful modeling of the natural phenomenon of code-switching in bilingual populations, and not as a tool for language teaching. While useful in generating coherent, syntactically constrained code-switched texts in its own right, none of these methods explicitly consider code-switching as a vehicle for teaching language, and thus do not take on an optimization-based view with an objective of improving lexical acquisition through the reading of the generated text. More recently, and concurrently with our work, Google's Language Immersion app employs the principle of

code-switching for language pedagogy, by generating code-switched web content, and allowing its users to tune it to their skill level. It does not, however, seem to model the user explicitly, nor is it clear if it performs any optimization in generating the text, as no studies have been published to date.

2.3 Computational Approaches to Sentence Simplification

Although not explicitly for teaching language, computational approaches that facilitate accessibility to texts that might otherwise be too difficult for its readers, either due to physical or learning disabilities, or language barriers, are relevant. In the recent work of (Kauchak, 2013), for example demonstrates an approach to increasing readability of texts by learning from unsimplified texts. Approaches in this area span methods for simplifying lexis (Yatskar et al., 2010; Biran et al., 2011), syntax (Siddharthan, 2006; Siddharthan et al., 2004), discourse properties (Hutchinson, 2005), and making technical terminology more accessible to non-experts (Elhadad and Sutaria, 2007). While the resulting texts are of great potential aid to language learners and may implicitly improve upon a reader's language proficiency, they do not explicitly attempt to promote learning as an objective in generating the simplified text.

2.4 Recent Neurophysiological findings

Evidence for the potential effectiveness of code-switching for language acquisition, stem from the recent findings of (Borovsky et al., 2012), who have shown that even a single exposure to a novel word in a constrained context, results in the integration of the word within your existing semantic base, as indicated by a change in the N400 electrophysiological response recorded from the subjects' scalps. N400 ERP marker has been found to correlate with the semantic "expectedness" of a word (Kutas and Hillyard, 1984), and is believed to be an early indicator of word learning. Furthermore, recent work of (Frank et al., 2013), show that word surprisal predicts N400, providing concrete motivation for artificial manipulation of text to explicitly elicit word learning through natural reading, directly motivating our approach. Prior to the above findings, it was widely believed that for evoking "incidental" word learning through reading alone, the word must appear with sufficiently high frequency within the text, such as to elicit the "noticing" effect — a prerequisite to lexical acquisition (Schmidt and Schmidt, 1995; Cobb, 2007).

3 Model

3.1 Overview

The formulation of our model is primarily motivated by two hypotheses that have been validated experimentally in the cognitive science literature. We re-state these hypotheses in the language of "surprisal":

1. Inserting a *target* word into a **low surprisal** context increases the rate of that word's integration into a learner's lexicon.

2. Multiple exposures to the word in **low surprisal** contexts increases rate of that word's integration.

Hypothesis 1 is supported by evidence from (Borovsky et al., 2012; Frank et al., 2013), and hypothesis 2 is supported by evidence from (Schmidt and Schmidt, 1995). We adopt the term "low-surprisal" context to identify contexts (e.g. n-grams) that are highly predictive of the target word (e.g. trailing word in the n-gram). The motivation stems from the recent evidence (Frank et al., 2013) that low-surprisal contexts affect the N400 response and thus correlate with word acquisition. To realize a "code-switched" mixture that adheres maximally to the above postulates, it is self-evident that a non-trivial optimization problem must be solved. For example, naively selecting a few words that appear in low-surprisal contexts may facilitate their acquisition, but at the expense of other words within the same context that may appear in a larger number of low-surprisal contexts further in the text.

To address this problem, we approach it with a formulation of a factor graph that takes global structure of the text into account. Factor graph formalism allows us to capture local features of individual contexts, such as lexical and syntactic surprisal, while inducing dependencies between consequent "switching decisions" in the text. Maximizing likelihood of the joint probability under the factorization of this graph yields an optimal sequence of these "switching decisions" in the entirety of the text. Maximizing joint likelihood, as we will show in the next section, is a surrogate to maximizing the probability of the learner acquiring novel words through the process of reading the generated text.

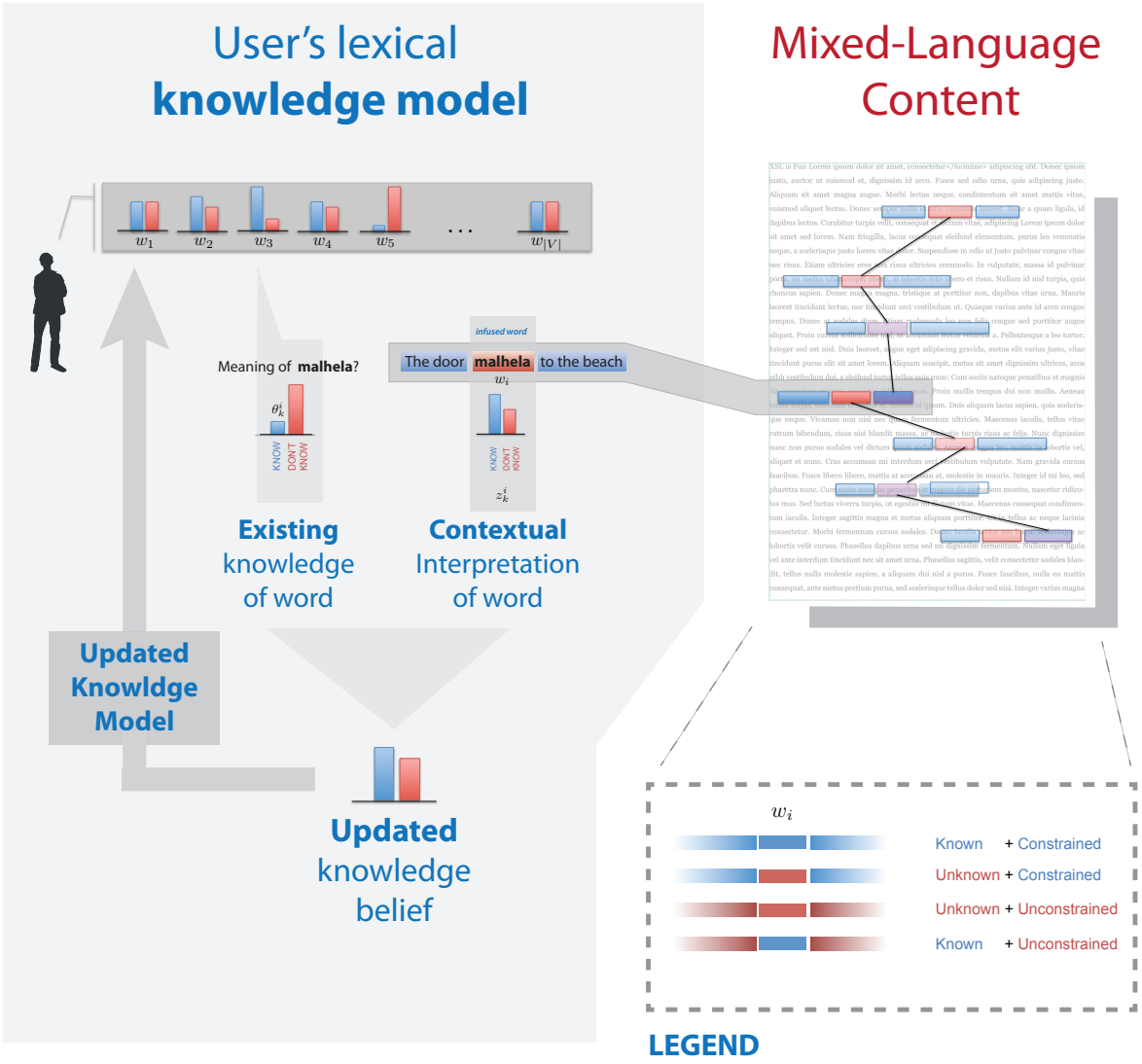


Figure 1: Overview of the approach. Probabilistic learner model (PLM) provides the current value of the belief in the learner's knowledge of any given word. Local contextual model provides the value of the belief in learning the word from the context alone. Upon exposure of the learner to the word in the given context, PLM is updated with the posterior belief in the user's knowledge of the word.

3.2 Language Learner Model

A simplified model of the learner, that we shall term a *Probabilistic Learner Model* (PLM) serves as a basis for our approach. PLM is a model of a learner's lexical knowledge at any given time. PLM models the learner as a vector of independent Bernoulli distributions, where each component represents a probability of the learner knowing the corresponding word. We motivate a probabilistic approach by taking the perspective of measuring *our* belief in the learner's knowledge of any given word, rather than the learner's uncertainty in own knowledge. Formally, we can fully specify this model for learner i as follows:

$$U_i = (\pi_0^i, \pi_1^i, \ldots, \pi_{|V|}^i) \quad (1)$$

where V is the vocabulary set — identical across all users, and π_j^i is our degree of belief in the learner i's knowledge of a target word $w_j \in V$. Statistical estimation techniques exist for estimating an individual's vocabulary size, such as (Bhat and Sproat, 2009; Beglar, 2010), and can be di-

rectly employed for estimating the parameters of this model as our prior belief about user i's knowledge.

The primary motivation behind a probabilistic user model, is to provide a mechanism for updating these probabilities as the user progresses through her reading. Maximizing the parameters of the PLM under a given finite span of code-switched text, thus, provides a handle for generating optimal code-switched content. Additionally, a probabilistic approach allows for a natural integration of the user model with the uncertainty in other components of the system, such as uncertainty in determining the degree of constraint imposed by the context, and in bitext alignment.

3.3 Model overview

At the high level, as illustrated in Figure 1, our approach integrates the model of the learner (PLM) with the local contextual features to update the PLM parameters incrementally as the learner progresses through the text. The fundamental assumption behind our approach is that the learner's knowledge of a given word *after* observing it in a sentence is a function of 1) the learner's previous knowledge of the word, *prior* to observing it in a given sentence and 2) a degree of constraint that a given context imposes on the meaning of the novel word, and is directly related to the surprisal of novel word in that context. Broadly, as the learner progresses from one sentence to the next, exposing herself to more novel words, the updated parameters of the language model in turn guide the selection of new "switch-points" for replacing source words with the target foreign words. In practice, however, this process is carried out implicitly and off-line by optimizing the estimated progress of the learner's PLM, without dynamic feedback. Next, we describe the model in detail.

3.4 Switching Factor Graph Model

To aid in the specification of the factor graph structure, we introduce new terminology. Because the PLM is updated progressively, we will refer to the parameters of the PLM for a given word w_i after observing its k^{th} appearance (instance) in the text, as the learner's *state* of knowledge of that word, and denote it as a binary random variable z_k^i.

$$P(z_k^i = 1) = \begin{cases} \text{Probability that} \\ \text{word } w_i \in V \\ \text{is understood on } k^{th}\text{exposure} \end{cases}$$

Without explicit testing of the user, this variable is hidden. We can view the prior learning model as the parameters of the vector of random variables $(z_0^0, z_0^1, \ldots z_0^{|V|})$.

The key to our approach is in how the parameters of these hidden variables are updated from repeated exposures to words in various contexts. Intuitively, an update to the parameter of z_k^i from z_{k-1}^i occurs after the learner observes word w_i in a context (this may be an n-gram, an entire sentence or paragraph containing w_i, but we will restrict our attention to fixed-length n-grams). Intuitively an update to the parameter of z_{k-1}^i will depend on how "constrained" the meaning of w_i is in the given context. We will refer to it as the "learnability", denoted by L_i^k, of word w_i on its k^{th} appearance, given its context. Formally, we will define "learnability" as follows:

$$P(L_k^i = 1 | w^i, \mathbf{w}^{\setminus i}, \mathbf{z}_k^{\setminus i}) = P(\text{constrained}(w_i) = 1 | \mathbf{w}) \prod_{i \neq j} P(z_k^j = 1) \quad (2)$$

where $\mathbf{w}^{\setminus i}$ represents the set of words that comprise the context window of w_i, not including w_i, and $\mathbf{z}_k^{\setminus i}$ are the states corresponding to each of the words in $\mathbf{w}^{\setminus i}$. $P(\text{constrained}(w_i) = 1 | \mathbf{w})$ is a real value (scaled between 0 and 1) that represents the degree of constraint imposed on the meaning of word w_i by its context. This value comes from a binary prediction model trained to predict the "predictability" of a word in its context, and is based on the dataset that we collected (described later in the paper). Generally, this value may come directly from the surprisal quantity given by a language model, or may incorporate additional features that are found informative in predicting the constraint on the word. Finally, the quantity is weighted by the parameters of the state variables corresponding to the words other than w_i contained in the context. This encodes an intuition that a degree of predictability of a given word given its context is related to the learner's knowledge of the other words in that context. If, for example, in the sentence "pass me the salt and pepper, please", both "salt" and "pepper" are substituted with their foreign translations that the learner is unlikely to know, it's equally unlikely that she will learn them after being exposed to this context, as the context itself will not offer sufficient

information for both words to be inferred simultaneously. On the other hand, substituting "salt" and "pepper" individually, is likely to make it much easier to infer the meaning of the other.

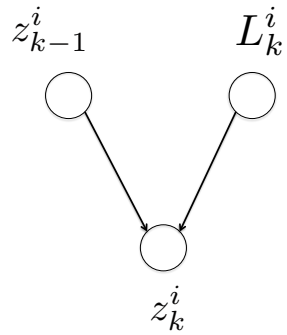


Figure 2: A noisy-OR combination of the learner's previous state of knowledge of the word z^i_{k-1} and the word's "learnability" in the observed context L^i_k

The updated parameter of z^i_k is obtained from a noisy-OR combination of the parameters of z^i_{k-1} and L^i_k:

$$P(z^i_k = 1 | z^i_{k-1}, L^i_k) = \\ 1 - [1 - P(L^i_k = 1)][1 - P(z_{k-1} = 1)]$$

A noisy-OR-based CPD provides a convenient and tractable approximation in capturing the intended intuition: updated state of knowledge of a given word will increase if the word is observed in a "good" context, or if the learner already knows the word.

Combining Equation 2 for each word in the context using the noisy-OR, the updated state for word w_i will now be conditioned on $z^i_{k-1}, \mathbf{z}^{\setminus i}_k, \mathbf{w}_k$. Because of the dependence of each z in the context on all other hidden variables in that context, we can capture the dependence using a single factor per context, with all of the z variables taking part in a clique, whose dimension is the size of the context.

We will now introduce a dual interpretation of the z variables: as "switching" variables that decide whether a given word will be replaced with its translation in the foreign language. If, for example, all of the words have high probability of being known by a learner, than maximizing the joint likelihood of the model will result in most of the words "switched-out" — a desired result. For an arbitrary prior PLM and the input text, maximizing joint likelihood will result in the selection of "switched-out" words that have the highest final probability of being "known" by the learner.

3.5 Inference

The problem of selecting "switch-points" reduces to the problem of inference in the resulting factor graph. Unfortunately, without a fairly strong constraint on the collocation of switched words, the resulting graph will contain loops, requiring techniques of approximate inference. To find the optimal settings of the z variables, we apply the loopy max-sum algorithm. While variants of loopy belief propagation, in general, are not guaranteed to converge, we found that the convergence does indeed occur in our experiments.

3.6 Predicting "predictable" words

We carried out experiments to determine which words are likely to be inferred from their context. The collected data-set is then used to train a logistic regression classifier to predict which words are likely to be easily inferred from their context. We believe that this dataset may also be useful to researchers in studying related phenomena, and thus make it publicly available.

For this task, we focus only on the following context features for predicting the "predictability" of words: n-gram probability, vector-space similarity score, coreferring mentions. N-gram probability and vector-space similarity [2] score are all computed within a fixed-size window of the word (trigrams using Microsoft N-gram service). *Coreference* feature is a binary feature which indicates whether the word has a co-referring mention in a 3-sentence window preceding a given context (obtained using Stanford's CoreNLP package). We train L2-regularized logistic regression to predict a binary label $L \in \{\text{Constrained}, \text{Unconstrained}\}$ using a crowd-sourced corpus described below.

3.7 Corpus Construction

For collecting data about which words are likely to be "predicted" given their content, we developed an Amazon Mechanical Turk task that presented turkers with excerpts of a short story (English translation of "The Man who Repented" by

[2] we employ C&W word embeddings from `http://metaoptimize.com/projects/wordreprs/`

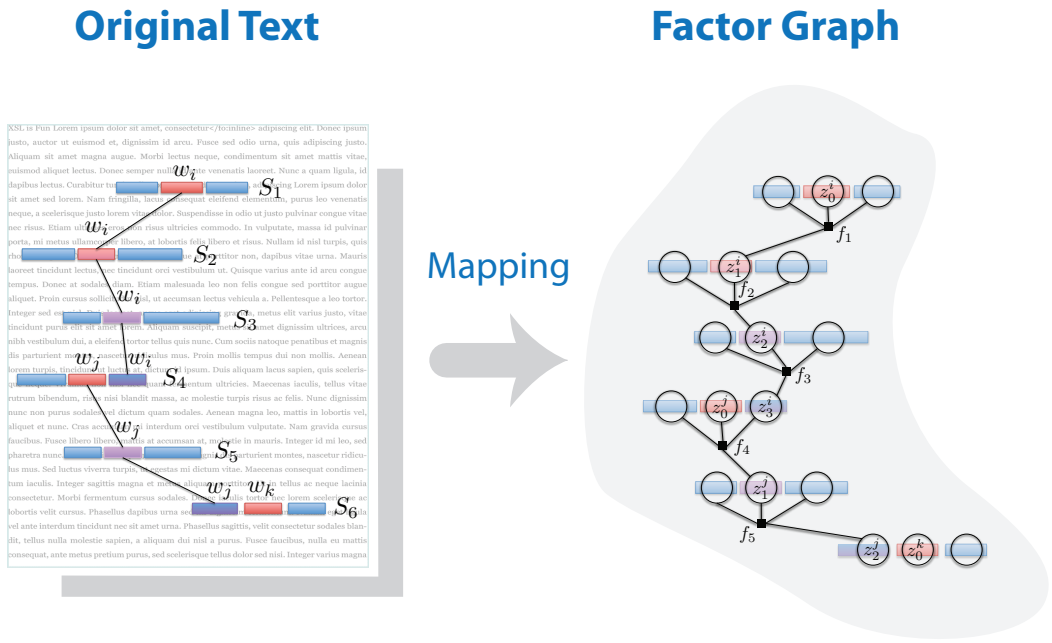


Figure 3: Sequence of sentences in the text (left) is mapped into a factor graph, whose nodes correspond to specific occurences of individual words, connected in a clique corresponding to a context in which the word occurs.

Ana Maria Matute), with some sentences containing a blank in place of a word. Only content words were considered for the task. Turkers were required to type in their best guess, and the number of semantically similar guesses were counted by an average number of 6 other turkers. A ratio of the median of semantically similar guesses to the total number of guesses was then taken as the score representing "predictability" of the word being guessed in the given context. All words corresponding to blanks whose scores were equal to and above 0.6 were than taken as a positive label (Constrained) and scores below 0.6 were taken as a negative label (Unconstrained). Turkers that judged the semantic similarity of the guesses of other turkers achieved an average Cohen's kappa agreement of 0.44, indicating fair to poor agreement.

4 Experiments

We carried out experiments on the effectiveness of our approach using the Amazon Mechanical Turk platform. Our experimental procedure was as follows: 162 turkers were partitioned into four groups, each corresponding to a treatment condition: OPT (N=34), HF (N=41), $RANDOM$ (N=43), MAN (N=44). Each condition corresponded to a model used to generate the presented code-switched text. For all experiments, the text used was a short story "Lottery" by Shirley Jackson, and a total number of replaced words was controlled (34). Target vocabulary consisted of words from an artificial language, generated statically by a mix of words from several languages. Below we describe the individual treatment conditions:

The café was dark and pokey.

The front overlooked the road and the rear the beach.

The door leading to the beach was hung with a reed curtain swaying in the breeze.

With every gust of wind it crackled a little like a slight rattle of bones.

Old Tomeu was sitting on the doorstep, leisurely stroking a well-worn tobacco pouch of black leather.

Figure 4: Visualization of the most "predictable" words in an excerpt from the "The Man who Repented" by Ana Maria Matute (English translation). Font-size correlates with the score given by judge turkers in evaluating guesses of other turkers that were presented with the same text, but the word replaced with a blank. Snippet of the dataset that we release publicly.

RANDOM (Baseline): words for switching are

selected at random from content only words.

HF (*High Frequency*) Baseline: words for switching are selected at random from a ranked list of words that occur most frequently in the presented text.

MAN (*Manual*) Baseline: words for switching are selected manually by the author, based on the intuition of which words are most likely to be guessed in context.

OPT (*Optimization-based*): factor graph-based model proposed in this paper is used for generating code-switched content. The total number of switched words generated by this method is used as a constant for all baselines.

Turkers were solicited to participate in a study that involved "reading a short story with a twist" (title of HIT). Not the title, nor the description gave away the purpose of the study, nor that it would be followed by a quiz. Time was not controlled for this study, but on average turkers took 27 minutes to complete the reading. Upon completing the reading portion of the task, turkers were presented with novel sentences that featured the words observed during reading, where only one of the sentences used the word in a semantically correct way. Turkers were asked to select the sentence that "made the most sense". An example of the sentences presented during the test:

> *Example 1*
>
> ✓ My edzino loves to go shopping every weekend.
>
> The edzino was too big to explore on our own, so went with a group.
>
> English word: **wife**
>
> *Example 2*
>
> ✓ His unpreadvers were utterly confusing and useless.
>
> The unpreadvers was so strong, that he had to go to a hospital.
>
> English word: **directions**

A "recall" metric was computed for each turker, defined as the ratio of correctly selected sentences to the total number of sentences presented. The "grand-average recall" across all turkers was then computed and reported here.

5 Results

We perform a one-way ANOVA across the four groups listed above, with the resulting $F = 11.38$ and $p = 9.7e{-}7$. Consequently, multiple pairwise comparison of the models was performed with the Bonferroni-corrected pairwise t-test, yielding the only significantly different recall means between $HF - MAN$ ($p = 0.00018$), $RANDOM - MAN$ ($p = 2.8e-6$), $RANDOM - OPT$ ($p = 0.00587$). The results indicate that, while none of the automated methods ($RANDOM$, HF, OPT) outperform manually generated code-switched text, OPT outperforms the $RANDOM$ baseline (no decisive conclusion can be drawn with respect to the $HF - RANDOM$ pair). Additionally, we note, that for words with frequency less than 4, OPT produces recall that is on average higher than the HF baseline (p=0.043, Welch's t-test), but at the expense of higher frequency words.

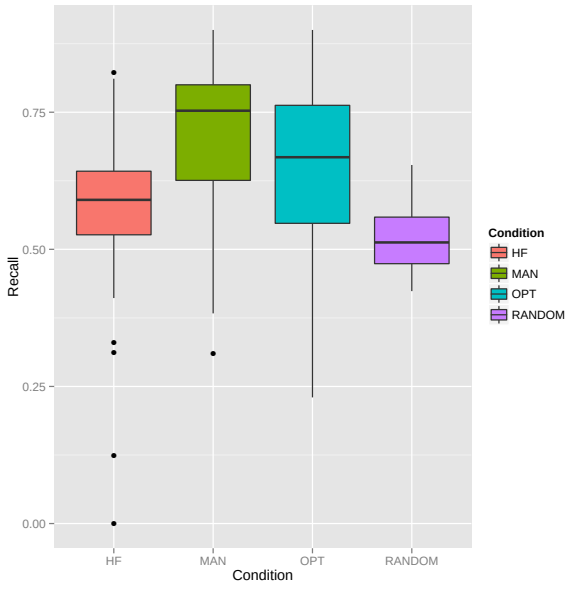


Figure 5: Results presented for 4 groups, subjected to 4 treatment conditions: $RANDOM$, HF, MAN, OPT. Recall performance for each group corresponds to the average ratio of selected sentences that correctly utilize code-switched words in novel contexts, across all turkers.

6 Discussion

We observe from our experiments that the optimization-based approach does not in general outperform the HF baseline. The strength of the

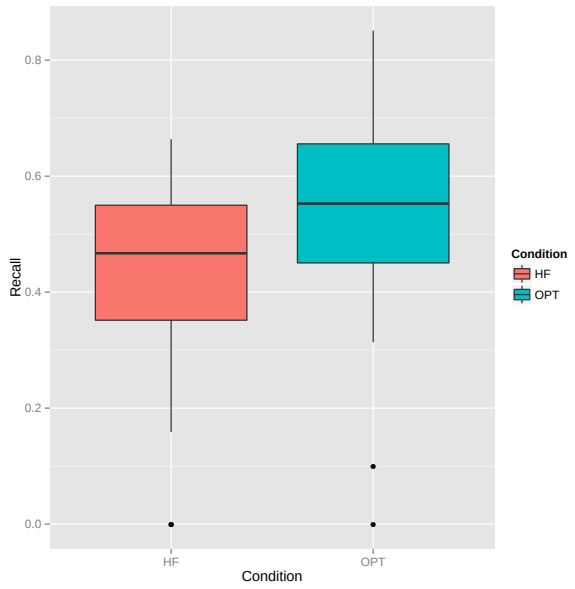


Figure 6: Subset of the results for 2 of the 4 treatment conditions: HF and OPT that correspond to recall only for words with item frequency in the presented text below 4.

frequency-based baseline is attributed to a well-known phenomenon that item frequency promotes the "noticing" effect during reading, critical for triggering incidental lexical acquisition. Generating code-switched text by replacing high frequency content words, thus, in general is a simple and viable approach for generating effective reading-based L2 curriculum aids. However, this method is fundamentally less flexible than the optimization-based method proposed in this paper, for several reasons:

- The optimization-based method explicitly models the learner and thus generates code-switched text progressively more fit for a given individual, even across a sequence of multiple texts. A frequency-based baseline alone would generate content at approximately the same level of difficulty consistently, with the pattern that words that tend to have high frequency in the natural language in general to be the ones that are "switched-out" most often.

- An optimization-based approach is able to elicit higher recall in low frequency words, as the mechanism for their selection is driven by the context in which these words appear, rather than frequency alone, favoring those that are learned more readily through context.

  Moreover, the proposed method in this paper is extensible to more sophisticated learner models, with a potential to surpass the results presented here. Another worthwhile application of this method is as a nested component within a larger optimization-based tool, that in addition to generating code-switched text as demonstrated here, aids in selecting content (such as popular books) as units in the code-switched curriculum.

7 Future Work

In this work we demonstrated a pilot implementation of a model-based, optimization-based approach to content generation for assisting in the reading-based L2 language acquisition. Our approach is based on static optimization, and while it would, in theory progress in difficulty with more reading, its open-loop nature precludes it from maintaining an accurate model of the learner in the long-term. For generating effecting L2 content, it is important that the user be kept in a "zone of proximal development" — a tight region where the level of the taught content is at just the right difficulty. Maintaining an accurate internal model of the learner is the single most important requirement for achieving this functionality. Closed-loop learning, with active user feedback is, thus, going to be functionally critical component of any system of this type that is designed to function in the long-term.

Additionally, our approach is currently a proof-of-concept of an automated method for generating content for assisted L2 acquisition, and is limited to artificial language and only isolated lexical items. The next step would be to integrate bitext alignment across texts in two natural languages, inevitably introducing another stochastic component into the pipeline. Extending this method to larger units, like chunks and simple grammar is another important avenue along which we are taking this work. Early results from concurrent research indicate that "code-switched based" method proposed here is also effective in eliciting acquisition of multi-word chunks.

References

Heike Adel, Ngoc Thang Vu, Franziska Kraus, Tim Schlippe, Haizhou Li, and Tanja Schultz. 2012. Re-

current neural network language modeling for code switching conversational speech. ICASSP.

David Beglar. 2010. A rasch-based validation of the vocabulary size test. *Language Testing*, 27(1):101–118.

Hedi M Belazi, Edward J Rubin, and Almeida Jacqueline Toribio. 1994. Code switching and x-bar theory: The functional head constraint. *Linguistic inquiry*, pages 221–237.

Suma Bhat and Richard Sproat. 2009. Knowing the unseen: estimating vocabulary size over unseen samples. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 109–117. Association for Computational Linguistics.

Rakesh Mohan Bhatt. 1997. Code-switching, constraints, and optimal grammars. *Lingua*, 102(4):223–251.

Or Biran, Samuel Brody, and Noemie Elhadad. 2011. Putting it simply: a context-aware approach to lexical simplification.

Fabian Blaicher. 2011. *SMT-based Text Generation for Code-Switching Language Models*. Ph.D. thesis, Nanyang Technological University, Singapore.

Arielle Borovsky, Jeffrey L Elman, and Marta Kutas. 2012. Once is enough: N400 indexes semantic integration of novel word meanings from a single exposure in context. *Language Learning and Development*, 8(3):278–302.

Tom Cobb. 2007. Computing the vocabulary demands of l2 reading. *Language Learning & Technology*, 11(3):38–63.

Noemie Elhadad and Komal Sutaria. 2007. Mining a lexicon of technical terms and lay equivalents. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*, pages 49–56. Association for Computational Linguistics.

Stefan L Frank, Leun J Otten, Giulia Galli, and Gabriella Vigliocco. 2013. Word surprisal predicts n400 amplitude during reading. In *Proceedings of the 51st annual meeting of the Association for Computational Linguistics*, pages 878–883.

Fred Genesee. 2001. Bilingual first language acquisition: Exploring the limits of the language faculty. *Annual Review of Applied Linguistics*, 21:153–168.

David R Hill. 2008. Graded readers in english. *ELT journal*, 62(2):184–204.

Ben Hutchinson. 2005. Modelling the substitutability of discourse connectives. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 149–156. Association for Computational Linguistics.

David Kauchak. 2013. Improving text simplification language modeling using unsimplified text data. In *Proceedings of ACL*.

Marta Kutas and Steven A Hillyard. 1984. Brain potentials during reading reflect word expectancy and semantic association. *Nature*.

James P Lantolf and Gabriela Appel. 1994. *Vygotskian approaches to second language research*. Greenwood Publishing Group.

John M Lipski. 2005. Code-switching or borrowing? no sé so no puedo decir, you know. In *Selected Proceedings of the Second Workshop on Spanish Sociolinguistics*, pages 1–15.

Ernesto Macaro. 2005. Codeswitching in the l2 classroom: A communication and learning strategy. In *Non-native language teachers*, pages 63–84. Springer.

Lesley Milroy and Pieter Muysken. 1995. *One speaker, two languages: Cross-disciplinary perspectives on code-switching*. Cambridge University Press.

Visvaganthie Moodley. 2010. Code-switching and communicative competence in the language classroom. *Journal for Language Teaching*, 44(1):7–22.

Ian SP Nation. 2001. *Learning vocabulary in another language*. Ernst Klett Sprachen.

Richard C Schmidt and Richard W Schmidt. 1995. *Attention and awareness in foreign language learning*, volume 9. Natl Foreign Lg Resource Ctr.

Norbert Schmitt, Diane Schmitt, and Caroline Clapham. 2001. Developing and exploring the behaviour of two new versions of the vocabulary levels test. *Language testing*, 18(1):55–88.

Advaith Siddharthan, Ani Nenkova, and Kathleen McKeown. 2004. Syntactic simplification for improving content selection in multi-document summarization. In *Proceedings of the 20th international conference on Computational Linguistics*, page 896. Association for Computational Linguistics.

Advaith Siddharthan. 2006. Syntactic simplification and text cohesion. *Research on Language and Computation*, 4(1):77–109.

Thamar Solorio and Yang Liu. 2008. Learning to predict code-switching points. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 973–981. Association for Computational Linguistics.

Mark Yatskar, Bo Pang, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2010. For the sake of simplicity: Unsupervised extraction of lexical simplifications from wikipedia. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 365–368. Association for Computational Linguistics.

Omni-word Feature and Soft Constraint for Chinese Relation Extraction

Yanping Chen[†] **Qinghua Zheng**[†] **Wei Zhang**[‡]
[†]MOEKLINNS Lab, Department of Computer Science and Technology
Xi'an Jiaotong University, China
ypench@gmail.com, qhzheng@mail.xjtu.edu.cn
[‡]Amazon.com, Inc.
wzhan@amazon.com

Abstract

Chinese is an ancient hieroglyphic. It is inattentive to structure. Therefore, segmenting and parsing Chinese are more difficult and less accurate. In this paper, we propose an Omni-word feature and a soft constraint method for Chinese relation extraction. The Omni-word feature uses every potential word in a sentence as lexicon feature, reducing errors caused by word segmentation. In order to utilize the structure information of a relation instance, we discuss how soft constraint can be used to capture the local dependency. Both Omni-word feature and soft constraint make a better use of sentence information and minimize the influences caused by Chinese word segmentation and parsing. We test these methods on the ACE 2005 RDC Chinese corpus. The results show a significant improvement in Chinese relation extraction, outperforming other methods in F-score by 10% in 6 relation types and 15% in 18 relation subtypes.

1 Introduction

Information Extraction (IE) aims at extracting syntactic or semantic units with concrete concepts or linguistic functions (Grishman, 2012; McCallum, 2005). Instead of dealing with the whole documents, focusing on designated information, most of the IE systems extract named entities, relations, quantifiers or events from sentences.

The relation recognition task is to find the relationships between two entities. Successful recognition of relation implies correctly detecting both the relation arguments and relation type. Although this task has received extensive research. The performance of relation extraction is still unsatisfactory with a F-score of 67.5% for English (23 subtypes) (Zhou et al., 2010). Chinese relation extraction also faces a weak performance having F-score about 66.6% in 18 subtypes (Dandan et al., 2012).

The difficulty of Chinese IE is that Chinese words are written next to each other without delimiter in between. Lacking of orthographic word makes Chinese word segmentation difficult. In Chinese, a single sentence often has several segmentation paths leading to the segmentation ambiguity problem (Liang, 1984). The lack of delimiter also causes the Out-of-Vocabulary problem (OOV, also known as *new word detection*) (Huang and Zhao, 2007). These problems are worsened by the fact that Chinese has a large number of characters and words. Currently, the state-of-the-art Chinese OOV recognition system has performance about 75% in recall (Zhong et al., 2012). The errors caused by segmentation and OOV will accumulate and propagate to subsequent processing (e.g. part-of-speech (POS) tagging or parsing).

Therefore, the Chinese relation extraction is more difficult. According to our survey, compared to the same work in English, the Chinese relation extraction researches make less significant progress.

Based on the characteristics of Chinese, in this paper, an Omni-word feature and a soft constraint method are proposed for Chinese relation extraction. We apply these approaches in a maximum entropy based system to extract relations from the ACE 2005 corpus. Experimental results show that our method has made a significant improvement.

The contributions of this paper include

1. Propose a novel Omni-word feature for Chinese relation extraction. Unlike the traditional segmentation based method, which is a partition of the sentence, the Omni-word feature uses every potential word in a sentence as lexicon feature.

2. Aiming at the Chinese inattentive structure, we utilize the soft constraint to capture the local dependency in a relation instance. Four constraint conditions are proposed to gener-

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 572–581,
Baltimore, Maryland, USA, June 23-25 2014.

ate combined features to capture the local dependency and maximize the classification determination.

The rest of this paper is organized as follows. Section 2 introduces the related work. The Omni-word feature and soft constrain are proposed in Section 3. We give the experimental results in Section 3.2 and analyze the performance in Section 4. Conclusions are given in Section 5.

2 Related Work

There are two paradigms extracting the relationship between two entities: the Open Relation Extraction (ORE) and the Traditional Relation Extraction (TRE) (Banko et al., 2008).

Based on massive and heterogeneous corpora, the ORE systems deal with millions or billions of documents. Even strict filtrations or constrains are employed to filter the redundancy information, they often generate tens of thousands of relations dynamically (Hoffmann et al., 2010). The practicability of ORE systems depends on the adequateness of information in a big corpus (Brin, 1999). Most of the ORE systems utilize weak supervision knowledge to guide the extracting process, such as: Databases (Craven and Kumlien, 1999), Wikipedia (Wu and Weld, 2007; Hoffmann et al., 2010), Regular expression (Brin, 1999; Agichtein and Gravano, 2000), Ontology (Carlson et al., 2010; Mohamed et al., 2011) or Knowledge Base extracted automatically from Internet (Mintz et al., 2009; Takamatsu et al., 2012). However, when iteratively coping with large heterogeneous data, the ORE systems suffer from the "semantic drift" problem, caused by error accumulation (Curran et al., 2007). Agichtein, Carlson and Fader et al. (2010; 2011; 2000) propose syntactic and semantic constraints to prevent this deficiency. The soft constraints, proposed in this paper, are combined features like these syntactic or semantic constraints, which will be discussed in Section 3.2.

The TRE paradigm takes hand-tagged examples as input, extracting predefined relation types (Banko et al., 2008). The TRE systems use techniques such as: Rules (Regulars, Patterns and Propositions) (Miller et al., 1998), Kernel method (Zhang et al., 2006b; Zelenko et al., 2003), Belief network (Roth and Yih, 2002), Linear programming (Roth and Yih, 2007), Maximum entropy (Kambhatla, 2004) or SVM (GuoDong et al., 2005). Compared to the ORE systems, the TRE systems have a robust performance. Disadvantages of the TRE systems are that the manually annotated corpus is required, which is time-consuming and costly in human labor. And migrating between different applications is difficult. However, the TRE systems are evaluable and comparable. Different systems running on the same corpus can be evaluated appropriately.

In the field of Chinese relation extraction, Liu et al. (2012) proposed a convolution tree kernel. Combining with external semantic resources, a better performance was achieved. Che et al. (2005) introduced a feature based method, which utilized lexicon information around entities and was evaluated on Winnow and SVM classifiers. Li and Zhang et al. (2008; 2008) explored the position feature between two entities. For each type of these relations, a SVM was trained and tested independently. Based on *Deep Belief Network*, Chen et al. (2010) proposed a model handling the high dimensional feature space. In addition, there are mixed models. For example, Lin et al. (2010) employed a model, combining both the feature based and the tree kernel based methods.

Despite the popularity of kernel based method, Huang et al. (2008) experimented with different kernel methods and inferred that simply migrating from English kernel methods can result in a bad performance in Chinese relation extraction. Chen and Li et al. (2008; 2010) also pointed out that, due to the inaccuracy of Chinese word segmentation and parsing, the tree kernel based approach is inappropriate for Chinese relation extraction. The reason of the tree kernel based approach not achieve the same level of accuracy as that from English may be that segmenting and parsing Chinese are more difficult and less accurate than processing English.

In our research, we proposed an Omni-word feature and a soft constraint method. Both approaches are based on the Chinese characteristics. Therefore, better performance is expected. In the following, we introduce the feature construction, which discusses the proposed two approaches.

3 Feature Construction

In this section, the employed candidate features are discussed. And four constraint conditions are proposed to transform the candidate features into combined features. The soft constraint is the

method to generate the combine features[1].

3.1 Candidate Feature Set

In the ACE corpus, an *entity* is an object or set of objects in the world. An *entity mention* is a reference to an entity. The entity mention is annotated with its full *extent* and its *head*, referred to as the *extend mention* and the *head mention* respectively. The extent mention includes both the head and its modifiers. Each *relation* has two entities as arguments: Arg-1 and Arg-2, referred to as E1 and E2. A *relation mention* (or instance) is the embodiment of a relation. It is referred by the sentence (or clause) in which the relation is located in. In our work, we focus on the detection and recognition of relation mention.

Relation identification is handled as a classification problem. Entity-related information (e.g. head noun, entity type, subtype, CLASS, LDC-TYPE, etc.) are supposed to be known and provided by the corpus. In our experiment, the entity type, subtype and the head noun are used.

All the employed features are simply classified into five categories: *Entity Type and Subtype*, *Head Noun*, *Position Feature*, *POS Tag* and *Omni-word Feature*. The first four are widely used. The last one is proposed in this paper and is discussed in detail.

Entity Type and Subtype: In ACE 2005 RDC Chinese corpus, there are 7 entity types (Person, Organization, GPE, Location, Facility, Weapon and Vehicle) and 44 subtypes (e.g. Group, Government, Continent, etc.).

Head Noun: The head noun (or head mention) of entity mention is manually annotated. This feature is useful and widely used.

Position Feature: The position structure between two entity mentions (extend mentions). Because the entity mentions can be nested, two entity mentions may have four coarse structures: "E1 is before E2", "E1 is after E2", "E1 nests in E2" and "E2 nests in E1", encoded as: '`E1_B_E2`', '`E1_A_E2`', '`E1_N_E2`' and '`E2_N_E1`'.

POS Tag: In our model, we use only the adjacent entity POS tags, which lie in two sides of the entity mention. These POS tags are labelled by the ICTCLAS package[2]. The POS tags are not used independently. It is encoded by combining the POS tag with the adjacent entity mention information. For example '`E1_Right_n`' means that the right side of the first entity is a noun ("n").

Omni-word Feature: The notion of "word" in Chinese is vague and has never played a role in the Chinese philological tradition (Sproat et al., 1996). Some Chinese segmentation performance has been reported precision scores above 95% (Peng et al., 2004; Xue, 2003; Zhang et al., 2003). However, for the same sentence, even native peoples in China often disagree on word boundaries (Hoosain, 1992; Yan et al., 2010). Sproat et al. (1996) has showed that there is a consistence of 75% on the segmentation among different native Chinese speakers. The word-formation of Chinese also implies that the meanings of a compound word are made up, usually, by the meanings of words that contained in it (Hu and Du, 2012). So, fragments of phrase are also informative.

Because high precision can be received by using simple lexical features (Kambhatla, 2004; Li et al., 2008). Making better use of such information is beneficial. In consideration of the Chinese characteristics, we use *every potential word in a relation mention* as the lexical features. For example, relation mention '台北大安森林公园' (Taipei Daan Forest Park) has a "PART-WHOLE" relation type. The traditional segmentation method may generate four lexical features {'台北', '大安', '森林', '公园'}, which is a partition of the relation mention. On the other hand, the Omni-word feature denoting all the possible words in the relation mention may generate features as:

> {'台', '北', '大', '安', '森', '林', '公', '园', '台北', '大安', '森林', '公园', '森林公园', '大安森林公园'}[3]

Most of these features are nested or overlapped mutually. So, the traditional character-based or word-based feature is only a subset of the Omni-word feature. To extract the Omni-word feature, only a lexicon is required, then scan the sentence to collect every word.

Because the number of lexicon entry determines the dimension of the feature space, performance of Omni-word feature is influenced by the lexicon being employed. In this paper, we generate the lexicon by merging two lexicons. The first lexicon

[1] If without ambiguity, we also use the terminology of "soft constraint" denoting features generated by the employed constraint conditions.

[2] `http://ictclas.org/`

[3] The generated Omni-word features dependent on the employed lexicon.

is obtained by segmenting every relation instance using the ICTCLAS package, collecting very word produced by ICTCLAS. Because the ICTCLAS package was trained on annotated corpus containing many meaningful lexicon entries. We expect this lexicon to improve the performance. The second lexicon is *the Lexicon Common Words in Contemporary Chinese*[4].

Despite the Omni-word can be seen as a subset of n-Gram feature. It is not the same as the n-Gram feature. N-Gram features are more fragmented. In most of the instances, the n-Gram features have no semantic meanings attached to them, thus have varied distributions. Furthermore, for a single Chinese word, occurrences of 4 characters are frequent. Even 7 or more characters are not rare. Because Chinese has plenty of characters[5], when the corpus becoming larger, the n-Gram (n¿4) method is difficult to be adopted. On the other hand, the Omni-word can avoid these problems and take advantages of Chinese characteristics (the word-formation and the ambiguity of word segmentation).

3.2 Soft Constraint

The structure information (or dependent information) of relation instance is critical for recognition. However, even in English, "deeper" analysis (e.g. logical syntactic relations or predicate-argument structure) may suffer from a worse performance caused by inaccurate chunking or parsing. Hence, the local dependency contexts around the relation arguments are more helpful (Zhao and Grishman, 2005). Zhang et al. (2006a) also showed that Path-enclosed Tree (PT) achieves the best performance in the kernel based relation extraction. In this field, the tree kernel based method commonly uses the parse tree to capture the structure information (Zelenko et al., 2003; Culotta and Sorensen, 2004). On the other hand, the feature based method usually uses the combined feature to capture such structure information (GuoDong et al., 2005; Kambhatla, 2004).

In the open relation extraction domain, syntactic and semantic constraints are widely employed to prevent the "semantic drift" problem. Such constraints can also be seen as structural constraint. Most of these constraints are hard constraints. Any relation instance violating these constraints (or below a predefined threshold) will be abandoned. For example, Agichtein and Gravano (2000) generates patterns according to a *confidence threshold* (τ_t). Fader et al. (2011) utilizes a *confidence function*. And Carlson et al. (2010) filters candidate instances and patterns using the number of times they co-occurs.

Deleting of relation instances is acceptable for open relation extraction because it always deals with a big data set. But it's not suitable for traditional relation extraction, and will result in a low recall. Utilizing the notion of combined feature (GuoDong et al., 2005; Kambhatla, 2004), we replace the hard constraint by the soft constraint. Each soft constraint (combined feature) has a parameter trained by the classifier indicating the discrimination ability it has. No subjective or priori judgement is adopted to delete any potential determinative constraint (except for the reason of dimensionality reduction).

Most of the researches make use of the combined feature, but rarely analyze the influence of the approaches we combine them. In this paper, we use the soft constraint to model the local dependency. It is a subset of the combined feature, generated by four constraint conditions: *singleton*, *position sensitive*, *bin sensitive* and *semantic pair* . For every employed candidate feature, an appropriate constraint condition is selected to combine them with additional information to maximize the classification determination.

Singleton: A feature is employed as a singleton feature when it is used without combining with any information. In our experiments, only the *position feature* is used as singleton feature.

Position Sensitive: A position sensitive feature has a label indicating which entity mention it depends on. In our experiment, the *Head noun* and *POS Tag* are utilized as position sensitive features, which has been introduced in Section 3.1. For example, '台北_E1' means that the head noun '台北' depend on the first entity mention.

Semantic Pair: Semantic pair is generated by combining two semantic units. Two kinds of semantic pair are employed. Those are generated by combining two entity types or two entity subtypes into a semantic pair. For example, 'Person_Location' denotes that the type of the first relation argument is a "Person" (entity

[4]Published by *Ministry of Education of the People's Republic of China* in 2008, containing 56,008 entries.

[5]Currently, at least 13000 characters are used by native Chinese people. *Modern Chinese Dictionary*: http://www.cp.com.cn/

type) and the second is a "Location" (entity type). Semantic pair can capture both the semantic and structure information in a relation mention.

Bin Sensitive: In our study, *Omni-word feature* is not added as "bag of words". To use the Omni-word feature, we segment each relation mention by two entity mentions. Together with the two entity mentions, we get five parts: "FIRST", "MIDDLE", "END", "E1" and "E2" (or less, if the two entity mentions are nested). Each part is taken as an independent bin. A flag is used to distinguish them. For example, '台北_Bin_F', '台北_Bin_E1' and '台北_Bin_E' mean that the lexicon entry '台北' appears in three bins: the FIRST bin, the first entity mention (E1) bin and the END bin. They will be used as three independent features.

To sum up, among the five candidate feature sets, the position feature is used as a singleton feature. Both head noun and POS tag are position sensitive. Entity types and subtypes are employed as semantic pair. Only Omni-word feature is bin sensitive. In the following experiments, focusing on Chinese relation extraction, we will analyze the performance of candidate feature sets and study the influence of the constraint conditions.

sectionExperiments

In this section, methodologies of the Omni-word feature and the soft constraint are tested. Then they are compared with the state-of-the-art methods.

3.3 Settings and Results

We use the ACE 2005 RDC Chinese corpus, which was collected from newswires, broadcasts and weblogs, containing 633 documents with 6 major relation types and 18 subtypes. There are 8,023 relations and 9,317 relation mentions. After deleting 5 documents containing wrong annotations[6], we keep 9,244 relation mentions as positive instances.

To get the negative instances, each document is segmented into sentences[7]. Those sentences that do not contain any entity mention pair are deleted. For each of the remained sentences, we iteratively extract every entity mention pair as the arguments of relation instances for predicting. For example, suppose a sentence has three entity mentions: A,B and C. Because the relation arguments are order sensitive, six entity mention pairs can be generated: [A,B], [A,C], [B,C], [B,A], [C,A] and [C,B]. After discarding the entity mention pairs that were used as positive instances, we generated 93,283 negative relation instances labelled as "OTHER". Then, we have 7 relation types and 19 subtypes.

A maximum entropy multi-class classifier is trained and tested on the generated relation instances. We adopt the five-fold cross validation for training and testing. Because we are interested in the 6 annotated major relation types and the 18 subtypes, we average the results of five runs on the 6 positive relation types (and 18 subtypes) as the final performance. F-score is computed by

$$\frac{2 \times (Precision \times Recall)}{Precision + Recall}$$

To implement the maximum entropy model, the toolkit provided by Le (2004) is employed. The iteration is set to 30.

Five candidate feature sets are employed to generate the combined features. The *entity type and subtype*, *head noun*, *position feature* are referred to as $\mathcal{F}_{thp}$[8]. The POS tags are referred to as $\mathcal{F}_{pos}$. The Omni-word feature set is denoted by $\mathcal{F}_{ow}$.

Table 1 gives the performance of our system on the 6 types and 18 subtypes. Note that, in this paper, bare numbers and numbers in the parentheses represent the results of the 6 types and the 18 subtypes respectively.

Table 1: Performance on Type (Subtype)

Features	P	R	F
$\mathcal{F}_{thp}$	61.51 (52.92)	48.85 (36.92)	54.46 (43.49)
$\mathcal{F}_{ow}$	80.16 (66.98)	75.45 (54.85)	77.74 (60.31)
$\mathcal{F}_{thp} \cup \mathcal{F}_{pos}$	83.93 (69.83)	77.81 (61.63)	80.76 (65.47)
$\mathcal{F}_{thp} \cup \mathcal{F}_{ow}$	92.40 (81.94)	88.37 (70.69)	90.34 (75.90)
$\mathcal{F}_{thp} \cup \mathcal{F}_{pos} \cup \mathcal{F}_{ow}$	92.26 (80.52)	88.51 (70.96)	90.35 (75.44)

In Row 1, because $\mathcal{F}_{thp}$ are features directly obtained from annotated corpus, we take this per-

[6] DAVYZW_{20041230.1024, 20050110.1403, 20050111.1514, 20050127.1720, 20050201.1538}.

[7] The five punctuations are used as sentence boundaries: Period (。), Question mark (？), Exclamatory mark (！), Semicolon (；) and Comma (，).

[8] "thp" is an acronym of "**t**ype, **h**ead, **p**osition". Features in $\mathcal{F}_{thp}$ are the candidate features combined with the corresponding constraint conditions. The following $\mathcal{F}_{pos}$ and $\mathcal{F}_{ow}$ are the same.

formance as our referential performance. In Row 2, with only the $\mathcal{F}_{ow}$ feature, the F-score already reaches 77.74% in 6 types and 60.31% in 18 subtypes. The last row shows that adding the $\mathcal{F}_{pos}$ almost has no effect on the performance when both the $\mathcal{F}_{thp}$ and $\mathcal{F}_{ow}$ are in use. The results show that $\mathcal{F}_{ow}$ is effective for Chinese relation extraction.

The superiorities of Owni-word feature depend on three reasons. First, the specificity of Chinese word-formation indicates that the subphrases of Chinese word (or phrase) are also informative. Second, most of relation instances have limited context. The Owni-word feature, utilizing every possible word in them, is a better way to capture more information. Third, the entity mentions are manually annotated. They can precisely segment the relation instance into corresponding bins. Segmentation of bins bears the sentence structure information. Therefore, the Owni-word feature with bin information can make a better use of both the syntactic information and the local dependency.

3.4 Comparison

Various systems were proposed for Chinese relation extraction. We mainly focus on systems trained and tested on the ACE corpus. Table 2 lists three systems.

Table 2: Survey of Other Systems

System	P	R	F
Che et al. (2005)	76.13	70.18	73.27
Zhang et al. (2011)	80.71 (77.75)	62.48 (60.20)	70.43 (67.86)
Liu et al. (2012)	81.1 (79.1)	61.0 (57.5)	69.0 (66.6)

Che et al. (2005) was implemented on the ACE 2004 corpus, with 2/3 data for training and 1/3 for testing. The performance was reported on 7 relation types: 6 major relation types and the none relation (or negative instance). Zhang et al. (2011) was based on the ACE 2005 corpus with 75% data for training and 25% for testing. Performances about the 7 types and 19 subtypes were given. Both of them are feature based methods. Liu et al. (2012) is a kernel based method evaluated on the ACE 2005 corpus. The five-fold cross validation was used and declared the performances on 6 relation types and 18 subtypes.

The data preprocessing makes differences from our experiments to others. In order to give a better comparison with the state-of-the-art methods, based on our experiment settings and data, we implement the two feature based methods proposed by Che et al. (2005) and Zhang et al. (2011) in Table 2. The results are shown in Table 3.

In Table 3, Ei ($i \in 1, 2$) represents entity mention. "Order" in Che et al. (2005) denotes the position structure of entity mention pair. Four types of order are employed (the same as ours). $Word_{Ei\pm k}$ and $POS_{Ei\pm k}$ are the words and POS of Ei, "$\pm k$" means that it is the kth word (of POS) after (+) or before (-) the corresponding entity mention. In this paper, $k = 1$ and $k = 2$ were set.

In Row 2, the "Uni-Gram" represents the Unigram features of internal and external character sequences. Internal character sequences are the four entity extend and head mentions. Five kinds of external character sequences are used: one In-Between character sequence between E1 and E2 and four character sequences around E1 and E2 in a given window size w_s. The w_s is set to 4. The "Bi-Gram" is the 2-gram feature of internal and external character sequences. Instead of the 4 position structures, the 9 position structures are used. Please refer to Zhang et al. (2011) for the details of these 9 position structures.

In Table 3, it is shown that our system outperforms other systems, in F-score, by 10% on 6 relation types and by 15% on 18 subtypes.

For researchers who are interested in our work, the source code of our system and our implementations of Che et al. (2005) and Zhang et al. (2011) are available at `https://github.com/YPench/CRDC`.

4 Discussion

In this section, we analyze the influences of employed feature sets and constraint conditions on the performances.

Most papers in relation extraction try to augment the number of employed features. In our experiment, we found that this does not always guarantee the best performance, despite the classifier being adopted is claimed to control these features independently. Because features may interact mutually in an indirect way, even with the same feature set, different constraint conditions can have significant influences on the final performance.

In Section 3, we introduced five candidate feature sets. Instead of using them as independent features, we combined them with additional in-

Table 3: Comparing With the State-of-the-Art Methods

System	Feature Set	P	R	F
(Che et al., 2005)	Ei.Type, Ei.Subtype, Order, $Word_{Ei\pm1}$, $Word_{Ei\pm2}$, $POS_{Ei\pm1}$, $POS_{Ei\pm2}$	84.81 (64.89)	75.69 (52.99)	79.99 (58.34)
(Zhang et al., 2011)	Ei.Type, Ei.Subtype, 9 Position Feature, Uni-Gram, Bi-Gram	79.56 (66.78)	72.99 (54.56)	76.13 (60.06)
Ours	$\mathcal{F}_{thp} \cup \mathcal{F}_{pos} \cup \mathcal{F}_{ow}$	92.26 (80.52)	88.51 (70.96)	90.35 (75.44)

formation. We proposed four constraint conditions to generate the soft constraint features. In Table 4, the performances of candidate features are compared when different constraint conditions was employed.

In Column 3 of Table 4 (**Constraint Condition**), (1), (2), (3), (4) and (5) stand for the referential feature sets[9] in Table 1. Symbol "/" means that the corresponding candidate features in the referential feature set are substituted by the new constraint condition. **Par** in Column 4 is the number of parameters in the trained maximum entropy model, which indicate the model complexity. **I** in Column 5 is the influence on performance. "-" and "+" mean that the performance is decreased or increased.

The first observation is that the combined features are more powerful than used as *singletons*. Model parameters are increased by the combined features. Increasing of parameters projects the relation extraction problem into a higher dimensional space, making the decision boundaries become more flexible.

The named entities in the ACE corpus are also annotated with the CLASS and LDCTYPE labels. Zhou et al. (2010) has shown that these labels can result in a weaker performance. Row 1, 2 and 3 show that, no matter how they are used, the performances decrease obviously. The reason of the performance degradation may be caused by the problem of over-fitting or data sparseness.

At most of the time, increase of model parameters can result in a better performance. Except in Row 8 and Row 11, when two *head nouns* of entity pair were combined as *semantic pair* and when *POS tag* were combined with the entity type, the performances are decreased. There are 7356 head nouns in the training set. Combining two head nouns may increase the feature space by $7356 \times (7356 - 1)$. Such a large feature space makes the occurrence of features close to a random distribution, leading to a worse data sparseness.

In Row 4, 10 and 13, these features are used as *singleton*, the performance degrades considerably. This means that, the missing of sentence structure information on the employed features can lead to a bad performance.

Row 9 and 12 show an interesting result. Comparing the reference set (5) with the reference set (3), the *Head noun* and *adjacent entity POS tag* get a better performance when used as *singletons*. These results reflect the interactions between different features. Discussion of this issue is beyond this paper's scope. In this paper, for a better demonstration of the constraint condition, we still use the *Position Sensitive* as the default setting to use the *Head noun* and the *adjacent entity POS tag*.

Row 13 and 14 compare the *Omni-word feature* (*By-Omni-word*) with the traditional segmentation based feature (*By-Segmentation*). *By-Segmentation* denotes the traditional segmentation based feature set generated by a segmentation tool, collecting every output of relation mention. In this place, the ICTCLAS package is adopted too.

Conventionally, if a sentence is perfectly segmented, *By-Segmentation* is straightforward and effective. But, our experiment shows different observations. Row 13 and 14 show that the *Omni-word* method outperforms the traditional method. Especially, when the *bin* information is used (Row 15), the performance of *Omni-word feature* increases considerably.

Row 14 shows that, compared with the traditional method, the Omni-word feature improves the performance by about 8.79% in 6 relation types and 11.83% in 18 subtypes in F-core. Such improvement may reside in the three reasons discussed in Section 3.3.

In short, from Table 4 we have seen that the *en-*

[9](1), (2), (3), (4) and (5) denote $\mathcal{F}_{thp}$, $\mathcal{F}_{ow}$, $\mathcal{F}_{thp} \cup \mathcal{F}_{pos}$, $\mathcal{F}_{thp} \cup \mathcal{F}_{ow}$ and $\mathcal{F}_{thp} \cup \mathcal{F}_{pos} \cup \mathcal{F}_{ow}$ respectively.

Table 4: Influence of Feature Set

No.	Feature	Constraint Condition	Par	P	R	F	I
1	entity CLASS and LDCTYPE	(1)/as singleton	21,112 21,910	60.29 (41.70)	42.82 (25.18)	50.07 (31.40)	-4.39 -12.09
2		(1)/combined with positional Info	21,159 22,013	63.02 (41.61)	44.47 (26.31)	52.15 (32.24)	-2.31 -11.25
3		(1)/as semantic pair	21,207 22,068	63.35 (42.98)	47.67 (31.34)	54.40 (36.25)	-0.06 -7.24
4	Type, Subtype semantic pair	(1)/as singleton	19,390 147,435	51.37 (32.8)	29.16 (18.97)	37.20 (24.06)	-17.26 -19.43
5		(1)/combined with positional info	19,524 20,297	61.77 (41.13)	43.67 (26.83)	51.17 (32.47)	-3.29 -11.02
6		(5)/as singleton	105,865 121,218	91.39 (79.32)	87.92 (68.73)	89.62 (73.65)	-0.73 -1.79
7	head noun	(3)/as singleton	21,450 22,409	85.66 (64.38)	75.74 (57.14)	80.40 (60.55)	-0.36 -0.34
8		(3)/as semantic pair	77,333 77,947	83.05 (59.70)	73.14 (51.70)	77.78 (55.41)	-2.54 -5.48
9		(5)/as singleton	100,963 115,499	92.50 (82.63)	88.90 (71.67)	90.66 (76.76)	+0.31 +1.32
10	adjacent entity POS tag	(3)/as singleton	21,450 22,409	72.66 (62.42)	61.16 (45.69)	66.41 (52.76)	-13.91 -8.13
11		(3)/combined with entity type	22,151 23,357	80.66 (63.41)	71.67 (53.16)	75.90 (57.83)	-4.42 -3.06
12		(5)/as singleton	106,931 121,194	92.50 (82.04)	88.66 (71.36)	90.54 (76.33)	+0.19 +0.89
13	Omni-word feature	(2)/*By-Segmentation* as singleton	36,916 41,652	67.19 (55.85)	60.12 (44.50)	63.46 (49.54)	-14.28 -10.77
14		(2)/*By-Segmentation* with bins	79,430 84,715	71.12 (54.76)	66.90 (43.50)	68.95 (48.48)	-8.79 -11.83
15		(2)/*By-Omni-word* as singleton	47,428 57,702	69.67 (54.85)	63.77 (48.84)	66.59 (51.67)	-11.15 -8.64
16		(5)/as singleton	57,321 67,722	91.43 (76.43)	86.37 (69.57)	88.83 (72.84)	-1.52 -2.60

tity type and subtype maximize the performance when used as *semantic pair*. *Head noun* and *adjacent entity POS tag* are employed to combine with positional information. *Omni-word feature* with bins information can increase the performance considerably. Our model (in Section 3.3) uses these settings. This insures that the performances of the candidate features are optimized.

5 Conclusion

In this paper, We proposed a novel Omni-word feature taking advantages of Chinese sub-phrases. We also introduced the soft constraint method for Chinese relation recognition. The soft constraint utilizes four constraint conditions to catch the structure information in a relation instance. Both the Omni-word feature and soft constrain make better use of information a sentence has, and minimize the deficiency caused by Chinese segmentation and parsing.

The size of the employed lexicon determines the dimension of the feature space. The first impression is that more lexicon entries result in more power. However, more lexicon entries also increase the computational complexity and bring in noises. In our future work, we will study this issue. The notion of soft constraints can also be extended to include more patterns, rules, regexes or syntac-

tic constraints that have been used for information extraction. The usability of these strategies is also left for future work.

Acknowledgments

The research was supported in part by NSF of China (91118005, 91218301, 61221063); 863 Program of China (2012AA011003); Cheung Kong Scholar's Program; Pillar Program of NST (2012BAH16F02); Ministry of Education of China Humanities and Social Sciences Project (12YJC880117); The Ministry of Education Innovation Research Team (IRT13035).

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of DL '00*, pages 85–94. ACM.

Michele Banko, Oren Etzioni, and Turing Center. 2008. The tradeoffs between open and traditional relation extraction. *Proceedings of ACL-HLT '08*, pages 28–36.

Sergey Brin. 1999. Extracting patterns and relations from the world wide web. *The World Wide Web and Databases*, pages 172–183.

Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka Jr, and Tom M. Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *Proceedings of WSDM '10*, pages 101–110.

Wanxiang Che, Ting Liu, and Sheng Li. 2005. Automatic entity relation extraction. *Journal of Chinese Information Processing*, 19(2):1–6.

Yu Chen, Wenjie Li, Yan Liu, Dequan Zheng, and Tiejun Zhao. 2010. Exploring deep belief network for chinese relation extraction. In *Proceedings of CLP '10*, pages 28–29.

Mark Craven and Johan Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of ISMB '99*, pages 77–86.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of ACL '04*, page 423.

James R. Curran, Tara Murphy, and Bernhard Scholz. 2007. Minimising semantic drift with mutual exclusion bootstrapping. In *Proceedings of PACLING '07*, pages 172–180.

Liu Dandan, Hu Yanan, and Qian Longhua. 2012. Exploiting lexical semantic resource for tree kernel-based chinese relation extraction. *Natural Language Processing and Chinese Computing*, pages 213–224.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of EMNLP '11*, pages 1535–1545.

Ralph Grishman. 2012. Information extraction: Capabilities and challenges. *Notes prepared for the*.

Zhou GuoDong, Su Jian, Zhang Jie, and Zhang Min. 2005. Exploring various knowledge in relation extraction. In *Proceedings of ACL '05*, pages 427–434.

Raphael Hoffmann, Congle Zhang, and Daniel S. Weld. 2010. Learning 5000 relational extractors. In *Proceedings of ACL '10*, volume 10, pages 286–295.

Rumjahn Hoosain. 1992. Psychological reality of the word in chinese. *Advances in psychology*, 90:111–130.

He Hu and Xiaoyong Du. 2012. Radical features for chinese text classification. In *Proceedings of FSKD '12*, pages 720–724.

Changning Huang and Hai Zhao. 2007. Chinese word segmentation : A decade review. *Journal of Chinese Information Processing*, 21(3):8–19.

Ruihong Huang, Le Sun, and Yuanyong Feng. 2008. Study of kernel-based methods for chinese relation extraction. *Information Retrieval Technology*, pages 598–604.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of ACL-demo '04*, page 22.

Zhang Le. 2004. Maximum entropy modeling toolkit for python and c++. *Natural Language Processing Lab, Northeastern University, China*.

Wenjie Li, Peng Zhang, Furu Wei, Yuexian Hou, and Qin Lu. 2008. A novel feature-based approach to chinese entity relation extraction. In *Proceedings of HLT-Short '08*, pages 89–92.

Nanyuan Liang. 1984. Written chinese word segmentation system-cdws. *Journal of Beijing Institute of Aeronautics and Astronautics*, 4.

Ruqi Lin, Jinxiu Chen, Xiaofang Yang, and Honglei Xu. 2010. Research on mixed model-based chinese relation extraction. In *Proceedings of ICCSIT '10*, volume 1, pages 687–691.

Andrew McCallum. 2005. Information extraction: Distilling structured data from unstructured text. *Queue*, 3(9):48–57.

Scott Miller, Michael Crystal, Heidi Fox, Lance Ramshaw, Richard Schwartz, Rebecca Stone, and Ralph Weischedel. 1998. Algorithms that learn to extract information: Bbn: Tipster phase iii. In *Proceedings of TIPSTER '98*, pages 75–89.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL '09*, pages 1003–1011.

Thahir P Mohamed, Estevam R Hruschka Jr., and Tom M Mitchell. 2011. Discovering relations between noun categories. In *Proceedings of EMNLP '11*, pages 1447–1455.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of COLING '04*.

Dan Roth and Wen-tau Yih. 2002. Probabilistic reasoning for entity & relation recognition. In *Proceedings of COLING '02*, pages 1–7.

Dan Roth and Wen-tau Yih. 2007. Global inference for entity and relation identification via a linear programming formulation. *Introduction to Statistical Relational Learning*, pages 553–580.

Richard Sproat, William Gale, Chilin Shih, and Nancy Chang. 1996. A stochastic finite-state word-segmentation algorithm for chinese. *Computational linguistics*, 22(3):377–404.

Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. 2012. Reducing wrong labels in distant supervision for relation extraction. In *Proceedings of ACL '12*, pages 721–729.

Fei Wu and Daniel S. Weld. 2007. Autonomously semantifying wikipedia. In *Proceedings of CIKM '07*, pages 41–50.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing*, 8(1):29–48.

Ming Yan, Reinhold Kliegl, Eike Richter, Antje Nuthmann, and Hua Shu. 2010. Flexible saccade-target selection in chinese reading. *The Quarterly Journal of Experimental Psychology*, 63(4):705–725.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *The Journal of Machine Learning Research*, 3:1083–1106.

Hua-Ping Zhang, Hong-Kui Yu, De-Yi Xiong, and Qun Liu. 2003. Hhmm-based chinese lexical analyzer ictclas. In *Proceedings of SIGHAN '03*, pages 184–187.

Min Zhang, Jie Zhang, and Jian Su. 2006a. Exploring syntactic features for relation extraction using a convolution tree kernel. In *Proceedings of HLT-NAACL '06*, pages 288–295.

Min Zhang, Jie Zhang, Jian Su, and Guodong Zhou. 2006b. A composite kernel to extract relations between entities with both flat and structured features. In *Proceedings of ACL '06*, pages 825–832.

Peng Zhang, Wenjie Li, Furu Wei, Qin Lu, and Yuexian Hou. 2008. Exploiting the role of position feature in chinese relation extraction. In *Proceedings of LREC '08*.

Peng Zhang, Wenjie Li, Yuexian Hou, and Dawei Song. 2011. Developing position structure-based framework for chinese entity relation extraction. *ACM Transactions on Asian Language Information Processing (TALIP)*, 10(3):14.

Shubin Zhao and Ralph Grishman. 2005. Extracting relations with integrated information using kernel methods. In *Proceedings of ACL '05*, pages 419–426.

Ming Zhong, Sheng Wang, and Ming Wu. 2012. Revising word lattice using support vector machine for chinese word segmentation. In *Proceedings of IIWAS '12*, pages 352–355.

Guodong Zhou, Longhua Qian, and Jianxi Fan. 2010. Tree kernel-based semantic relation extraction with rich syntactic and semantic information. *Information Sciences*, 180(8):1313–1325.

Bilingual Active Learning for Relation Classification via Pseudo Parallel Corpora

Longhua Qian Haotian Hui Ya'nan Hu Guodong Zhou* Qiaoming Zhu
Natural Language Processing Lab
School of Computer Science and Technology, Soochow University
1 Shizi Street, Suzhou, China 215006
{qianlonghua,20134227019,20114227025,gdzhou,qmzhu}@suda.edu.cn

Abstract

Active learning (AL) has been proven effective to reduce human annotation efforts in NLP. However, previous studies on AL are limited to applications in a single language. This paper proposes a bilingual active learning paradigm for relation classification, where the unlabeled instances are first jointly chosen in terms of their prediction uncertainty scores in two languages and then manually labeled by an oracle. Instead of using a parallel corpus, labeled and unlabeled instances in one language are translated into ones in the other language and all instances in both languages are then fed into a bilingual active learning engine as pseudo parallel corpora. Experimental results on the ACE RDC 2005 Chinese and English corpora show that bilingual active learning for relation classification significantly outperforms monolingual active learning.

1 Introduction

Semantic relation extraction between named entities (*aka*. entity relation extraction or more concisely relation extraction) is an important subtask of Information Extraction (IE) as well as Natural Language Processing (NLP). With its aim to identify and classify the semantic relationship between two entities (ACE 2002-2007), relation extraction is of great significance to many NLP applications, such as question answering, information fusion, social network construction, and knowledge mining and population etc.

In the literature, the mainstream research on relation extraction adopts statistical machine learning methods, which can be grouped into supervised learning (Zelenko et al., 2003; Culotta and Soresen, 2004; Zhou et al., 2005; Zhang et al., 2006; Qian et al., 2008; Chan and Roth, 2011), semi-supervised learning (Zhang et al., 2004; Chen et al., 2006; Zhou et al., 2008; Qian et al., 2010) and unsupervised learning (Hasegawa et al., 2004; Zhang et al., 2005) in terms of the amount of labeled training data they need. Usually the extraction performance depends heavily on the quality and quantity of the labeled data, however, the manual annotation of a large-scale corpus is labor-intensive and time-consuming. In the last decade researchers have turned to another effective learning paradigm--active learning (AL), which, given a small number of labeled instances and a large number of unlabeled instances, selects the most informative unlabeled instances to be manually annotated and add them into the training data in an iterative fashion. Essentially active learning attempts to decrease the quantity of labeled instances by enhancing their quality, gauged by their informativeness to the learner. Since its emergence, active learning has been successfully applied to many tasks in NLP (Engelson and Dagan, 1996; Hwa, 2004; Tomanek et al., 2007; Settles and Craven, 2008).

It is trivial to validate, as we will do later in this paper, that active learning can also alleviate the annotation burden for relation extraction in one language while retaining the extraction performance. However, there are cases when we may exploit relation extraction in multiple languages and there are corpora with relation instances annotated for more than one language, such as the ACE RDC 2005 English and Chinese corpora. Hu et al. (2013) shows that supervised relation extraction in one language (e.g. Chinese)

* Corresponding author

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 582–592,
Baltimore, Maryland, USA, June 23-25 2014.

can be enhanced by relation instances translated from another language (e.g. English). This demonstrates that there is some complementariness between relation instances in two languages, particularly when the training data is scarce. One natural question is: Can this characteristic be made full use of so that active learning can maximally benefit relation extraction in two languages? To the best of our knowledge, so far the issue of joint active learning in two languages has yet been addressed. Moreover, the success of joint bilingual learning may lend itself to many inherent multilingual NLP tasks such as POS tagging (Yarowsky and Ngai, 2001), name entity recognition (Yarowsky et al., 2001), sentiment analysis (Wan, 2009), and semantic role labeling (Sebastian and Lapata, 2009) etc.

This paper proposes a bilingual active learning (BAL) paradigm to relation classification with a small number of labeled relation instances and a large number of unlabeled instances in two languages (non-parallel). Instead of using a parallel corpus which should have entity/relation alignment information and is thus difficult to obtain, this paper employs an off-the-shelf machine translator to translate both labeled and unlabeled instances from one language into the other language, forming pseudo parallel corpora. These translated instances along with the original instances are then fed into a bilingual active learning engine. Findings obtained from experiments with relation classification on the ACE 2005 corpora show that this kind of pseudo-parallel corpora can significantly improve the classification performance for both languages in a BAL framework.

The rest of the paper is organized as follows. Section 2 reviews the previous work on relation extraction while Section 3 describes our baseline systems. Section 4 elaborates on the bilingual active learning paradigm and Section 5 discusses the experimental results. Finally conclusions and directions for future work are presented in Section 6.

2 Related Work

While there are many studies in monolingual relation extraction, there are only a few on multilingual relation extraction in the literature.

Monolingual relation extraction: A wide range of studies on relation extraction focus on monolingual resources. As far as representation of relation instances is concerned, there are feature-based methods (Zhao et al., 2004; Zhou et al., 2005; Chan and Roth, 2011) and kernel-based methods (Zelenko et al., 2003; Zhang et al., 2006; Qian et al., 2008), mainly for the English language. Both methods are also widely used in relation extraction in other languages, such as those in Chinese relation extraction (Che et al., 2005; Li et al., 2008; Yu et al., 2010).

Multilingual relation extraction: There are only two studies related to multilingual relation extraction. Kim et al. (2010) propose a cross-lingual annotation projection approach which uses parallel corpora to acquire a relation detector on the target language. However, the mapping of two entities involved in a relation instance may leads to errors. Therefore, Kim and Lee (2012) further employ a graph-based semi-supervised learning method, namely Label Propagation (LP), to indirectly propagate labels from the source language to the target language in an iterative fashion. Both studies transfer relation annotations via parallel corpora from the resource-rich language (English) to the resource-poor language (Korean), but not vice versa. Based on a small number of labeled instances and a large number of unlabeled instances in both languages, our method differs from theirs in that we adopt a bilingual active learning paradigm via machine translation and improve the performance for both languages simultaneously.

Active Learning in NLP: Active learning has become an active research topic due to its potential to significantly reduce the amount of labeled training data while achieving comparable performance with supervised learning. It has been successfully applied to many NLP applications, such as POS tagging (Engelson and Dagan, 1996; Ringger et al., 2007), word sense disambiguation (Chan and Ng, 2007; Zhu and Hovy, 2007), sentiment detection (Brew et al., 2010; Li et al., 2012), syntactical parsing (Hwa, 2004; Osborne and Baldridge, 2004), and named entity recognition (Shen et al., 2004; Tomanek et al., 2007; Tomanek and Hahn, 2009) etc.

Different from these AL studies on a single task, Reichart et al. (2008) introduce a multi-task active learning (MTAL) paradigm, where unlabeled instances are selected for two annotation tasks (i.e. named entity and syntactic parse tree). They demonstrate that MTAL in the same language outperforms one-sided and random selection AL. From a different perspective, we propose an active learning framework for the same task, but across two different languages.

Another related study (Haffari and Sarkar, 2009) deals with active learning for multilingual

machine translation, which make use of multilingual corpora to decrease human annotation efforts by selecting highly informative sentences for a newly added language in multilingual parallel corpora. While machine translation inherently deals with multilingual parallel corpora, our task focuses on relation extraction by pseudo parallel corpora in two languages.

3 Baseline Systems

This section first introduces the fundamental supervised learning method, and then describes a baseline active learning algorithm.

3.1 Supervised Learning

We adopt the feature-based method for fundamental supervised relation classification, rather than the tree kernel-based method, since active learning needs a large number of iterations and the kernel-based method usually performs much slower than the feature-based one. Following is a list of our used features, much similar to Zhou et al. (2005):

a) Lexical features of entities and their contexts
- WM1: bag-of-words in the 1st entity mention
- HM1: headword of M1
- WM2: bag-of-words in the 2nd entity mention
- HM2: headword of M2
- HM12: combination of HM1 and HM2
- WBNULL: when no word in between
- WBFL: the only one word in between
- WBF: the first word in between when at least two words in between
- WBL: the last word in between when at least two words in between
- WBO: other words in between except the first and last words when at least three words in between

b) Entity type
- ET12: combination of entity types
- EST12: combination of entity subtypes
- EC12: combination of entity classes

c) Mention level
- ML12: combination of entity mention levels
- MT12: combination of LDC mention types

d) Overlap
- #WB: number of other mentions in between
- #MB: number of words in between
- M1>M2 or M1<M2: flag indicating whether M2/M1 is included in M1/M2.

3.2 Active Learning Algorithm

We use a pool-based active learning procedure with uncertainty sampling (Scheffer et al., 2001; Culotta and McCallum, 2005; Kim et al., 2006) for both Chinese and English relation classification as illustrated in Fig. 1. During iterations a batch of unlabeled instances are chosen in terms of their informativeness to the current classifier, labeled by an oracle and in turn added into the labeled data to retrain the classifier. Due to our focus on the effectiveness of bilingual active learning on relation classification, we only use uncertainty sampling without incorporating more complex measures, such as diversity and representativeness (Settles and Craven, 2008), and leave them for future work.

Algorithm uncertainty-based active learning

Input:
- *L*, labeled data set
- *U*, unlabeled data set
- *n*, batch size

Output:
- *SVM*, classifier

Repeat:
1. Train a single classifier *SVM* on *L*
2. Run the classifier on *U*
3. Find at most *n* instances in *U* that the classifier has the highest prediction uncertainty
4. Have these instances labeled by an oracle
5. Add them into *L*

Until: certain number of instances are labeled or certain performance is reached

Figure 1. Pool-based active learning with uncertainty sampling

Since the SVMLIB package used in this paper can output probabilities assigned to the class labels on an instance, we have three uncertainty metrics readily available, i.e., least confidence (**LC**), margin (**M**) and entropy (**E**). The NER experimental results on multiple corpora (Settles and Craven, 2008) show that there is no single clear winner among these three metrics. This conclusion is also validated by our preliminary experiments on the task of active learning relation extraction, thus we adopt the **LC** metric for simplicity. Specifically, with a sequence of K probabilities for a relation instance at some iteration, denoted as $\{p_1,p_2,\ldots p_K\}$ in the descending order, the **LC** metric of the relation instance can be simply picked as the first one, i.e.

$$H^{LC} = p_1 \quad (1)$$

Where K denotes the total number of relation classes. Note that this metric actually reflects prediction reliability (i.e. reverse uncertainty) rather than uncertainty in order to facilitate joint

confidence calculation for two languages (cf. §4.4). Intuitively, the smaller the H^{LC} is, the less confident the prediction is.

4 Bilingual Active Learning for Relation Classification

In this section, we elaborate on the bilingual active learning for relation extraction.

4.1 Problem Definition

With Chinese and English (designated as *c* and *e*) as two languages used in our study, this paper intends to address the task of bilingual relation classification, i.e., assigning relation labels to candidate instances that have semantic relationships. Suppose we have a small number of labeled instances in both languages, denoted as L_c and L_e (non-parallel) respectively, and a large number of unlabeled instances in both languages, denoted as U_c and U_e (non-parallel). The test instances in both languages are represented as T_c and T_e. In order to take full advantage of bilingual resources, we translate both labeled and unlabeled instances in one language to ones in the other language as follows:

$L_c \rightarrow L_{et}$
$U_c \rightarrow U_{et}$
$L_e \rightarrow L_{ct}$
$U_e \rightarrow L_{ct}$

The objective is to learn SVM classifiers in both languages, denoted as SVM_c and SVM_e respectively, in a BAL fashion to improve their classification performance.

4.2 Bilingual Active Learning Framework

Currently, AL is widely used in NLP tasks in a single language, i.e., during iterations unlabeled instances least confident only in one language are picked and manually labeled to augment the training data. The only exception is AL for machine translation (Haffari et al., 2009; Haffari and Sarkar, 2009), whose purpose is to select the most informative sentences in the source language to be manually translated into the target language. Previous studies (Reichart et al., 2008; Haffari and Sarkar, 2009) show that multi-task active learning (MTAL) can yield promising overall results, no matter whether they are two different tasks or the task of machine translation on multiple language pairs. If a specific NLP task on two languages, such as relation classification, can be regarded as two tasks, it is reasonable to argue that these two tasks can benefit each other when jointly performed in the BAL framework. Yet, to our knowledge, this issue remains unexplored.

An important issue for bilingual learning is how to obtain two language views for relation instances from multilingual resources. There are three solutions to this problem, i.e. parallel corpora (Lu et al., 2011), translated corpora (*aka*. pseudo parallel corpora) (Wan 2009), and bilingual lexicons (Oh et al., 2009). We adopt the one with pseudo parallel corpora, using the machine translation method to generate instances from one language to the other in the BAL paradigm, as depicted in Fig. 2.

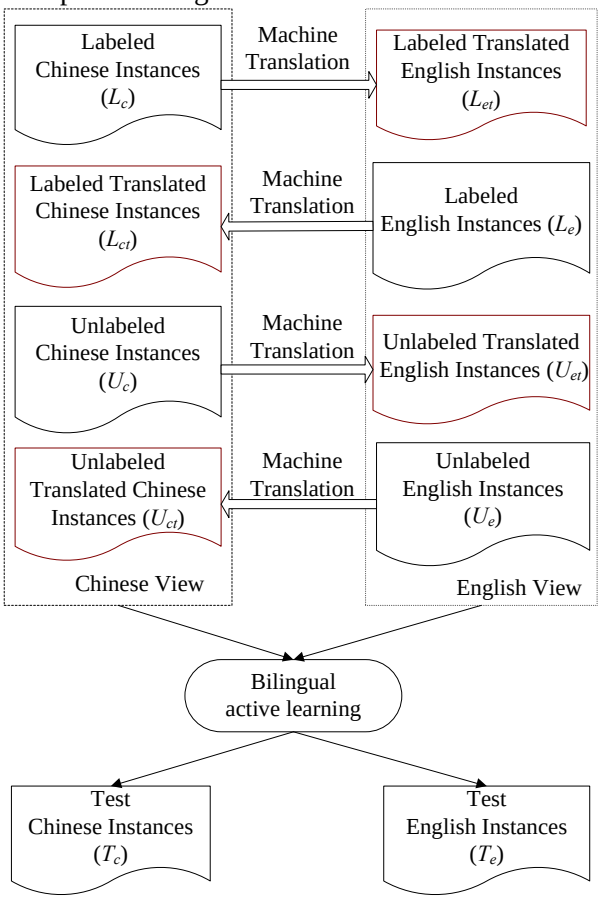


Figure 2. Framework of bilingual active learning

In order to make full use of pseudo parallel corpora, translated labeled and unlabeled instances are augmented in the following two ways:

- For labeled Chinese instances (L_c) and English instances (L_e), their translated counterparts (L_{et} and L_{ct}), along with their labels, are directly added into the labeled instances in the other language;
- For unlabeled Chinese instances (U_c) and English instances (U_e), during an active learning iteration the top *n* unlabeled instances in U_c and U_{et} which are least confidently jointly

predicted by SVM_c and SVM_e are labeled by an oracle and added to L_c and L_e respectively. (cf. §4.4)

4.3 Instance Projection via MT

Among the several off-the-shelf machine translation services, we select the Google Translator[1] because of its high quality and easy accessibility. Both the mentions of relation instances and the mentions of two involved entities are first translated into the other language via machine translation. Then, two entities in the original instance are aligned with their counterparts in the translated instance in order to form an aligned bilingual relation instance pair.

Instance translation

All the positive instances in the ACE 2005 Chinese and English corpora are translated to another language respectively, i.e. Chinese to English and vice versa. The relation instance is represented as the word sequence between two entities. This word sequence, rather than the whole sentence, is then translated to another language by the Google Translator. The reason is that, although this sequence loses partial contextual information of the relation instance, its translation quality is supposed to be better. Our preliminary experiments indicate that the addition of contextual information fail to benefit the task. After translation, word segmentation is performed on Chinese instances translated from English while tokenization is needed for translated English instances.

Entity alignment

The objective of entity alignment is to build a mapping from the entities in the original instances to the entities in the translated instances. Put in another way, entity alignment automatically marks the entity mentions in the translated instance, thereby the feature vector corresponding to the translated instance can be constructed. Entity alignment is vital in cross-language relation extraction whose difficulty lies in the fact that the same entity mention as an isolated phrase and as an integral phrase in the relation instance can be translated to different phrases. For example, the Chinese entity mention "官员" (*officer*) is translated to "officer" in isolation, it is, however, translated to "officials" when in the relation instance "叙利亚 官员" (*Syrian officials*).

[1] http://translate.google.com

Algorithm entity alignment

Input:

- M_e, entity mention in English
- R_e, relation instance in English
- M_{ct}, translation of M_e in Chinese
- R_c, translation of R_e in Chinese
- L, a lexicon consisting of entries like (e_i, c_i, p_i), where p_i is the translation probability from e_i to c_i
- α, probability threshold

Output:

- M_c, the counterpart of M_e in R_c

Steps:

1. If M_{ct} can be exactly found in R_c, then return M_{ct}
2. If the rightmost part of M_{ct} can be found in R_c, then this part can be returned
3. For very word w_e in M_e,
   a) If there exists a word w_c in R_c and (w_e, w_c, p) in L and $p>\alpha$, then (w_e, w_c) is a match of two words
   b) Return a successive sequence of matching words w_c
4. Return null

Figure 3. Entity alignment algorithm

Therefore, we devise some heuristics to align entity mentions between Chinese and English. The basic idea is that the word sequence in one mention successively matches the word sequence in the other mention. Take entity alignment from English to Chinese as an example, given entity mention M_e in relation instance R_e in English and their respective translations M_{ct} and R_c in Chinese, the objective of entity alignment is to find M_c, the counterpart of M_e in R_c. The procedure of entity alignment algorithm can be described in Fig. 3.

In the algorithm, the probability threshold α is empirically set to 0.002 where the precision and recall of entity alignment are balanced. Our lexicon is derived from the FBIS parallel corpus (#LDC2003E14), which is widely used in machine translation between English and Chinese. It should be noted that the process of relation translation and entity alignment are far from perfection, leading to reduction in the number of instances being mapping to the other language, i.e.

$$|L_c| > |L_{et}|$$
$$|U_c| > |U_{et}|$$
$$|L_e| > |L_{ct}|$$
$$|U_e| > |L_{ct}|$$

4.4 Bilingual Active Learning Algorithm

The basic idea of our BAL paradigm is that, while unlabeled instances uncertain in one lan-

guage are informative to the learner in that language, unlabeled instances jointly uncertain in both languages are informative to the learners in both languages, thus potentially improving classification performance for both languages more than their individual active learners do. This idea is embodied in the BAL algorithm in Fig. 4, where n is the batch size, i.e., the number of instances selected, labeled and augmented at each iteration.

Algorithm bilingual active learning

Input:

- L_c and U_c, labeled and unlabeled instances in Chinese, and L_{et} and U_{et}, their respective translation counterparts in English
- L_e and U_e, labeled and unlabeled instances in English, and L_{ct} and U_{ct}, their translation counterparts in Chinese
- n, batch size

Output:

- SVM_c and SVM_e, two classifiers for Chinese and English respectively

Initialize:

1. Add instances L_{ct} to L_c
2. Add instances L_{et} to L_e

Repeat:

1. Learn the Chinese classifier SVM_c from L_c
2. Use SVM_c to classify instances in U_c and U_{ct}
3. Learn the English classifier SVM_e from L_e
4. Use SVM_e to classify instances in U_e and U_{et}
5. Choose the n least confidently jointly predicted instance pairs $\{E_c|E_{et}\}$ from $\{U_c|U_{et}\}$, and have them labeled by an oracle
6. Choose the n least confidently jointly predicted instance pairs $\{E_e|E_{et}\}$ from $\{U_e|U_{ct}\}$, and have them labeled by an oracle
7. Remove E_c from U_c and E_e from U_e
8. Add instances $E_c \cup E_{ct}$ to L_c with their manual labels
9. Add instances $E_e \cup E_{et}$ to L_e with their manual labels

Until certain number of instances are labeled or certain performance is reached

Figure 4. Bilingual active learning algorithm

The key point of this algorithm lies in Step 5 and Step 6, where unlabeled instances from U_c and U_e are selected and labeled respectively. Take Chinese for an example, when gauging the prediction uncertainty for an unlabeled instance in U_c, not only its own uncertainty measure H_c predicted by SVM_c is considered, but also the uncertainty measure H_{et} for its translation counterpart in U_{et}, which is predicted by SVM_e, is considered. Generally, in order to jointly consider these two measures, there are three methods to compute their means, namely, arithmetic mean, geometric mean and harmonic mean. Preliminary experiments show that among these three means, there is no single winner, so we simply take the geometric mean defined as follows:

$$H_g = \sqrt{H_c * H_{et}} \quad (2)$$

Considering that we adopt the **LC** measure as the uncertainty score, when an instance in U_c can't find its translation counterpart in U_{et} due to translation error or entity alignment failure, H_{et} is set to 1, i.e. the maximum. Since the bigger H is, the more confident the prediction is, the less likely the instance will be chosen, in this way we discourage the unlabeled instances without translation counterparts.

5 Experimentation

We have systematically evaluated our BAL paradigm on the relation classification task using ACE RDC 2005 RDC Chinese and English corpora.

5.1 Experimental Settings

Corpora and Preprocessing

We use the ACE 2005 RDC Chinese and English corpora as the benchmark data (hereafter we refer to them as the Chinese corpus (ACE2005c) and the English corpus (ACE2005e) respectively). Both corpora have the same entity/relation hierarchies, which define 7 entity types, 6 major relation types. However, the Chinese corpus contains 633 documents and 9,147 positive relation instances while the English corpus only contains 498 files and 6,253 positive instances. Therefore, in order to balance the corpus scale to fairly evaluate bilingual active learning impact on relation classification, we randomly select 458 Chinese files and thus get 6,268 positive instances, comparable to the English corpus.

Preprocessing steps for both corpora include sentence splitting and tokenization (word segmentation for Chinese using ICTCLAS[2]). Then, positive relation mentions with word sequences between two entities and their feature vectors are extracted from sentences while negative relation mentions are simply discarded because we focus on the task of relation classification. After entity and relation mentions in one language are trans-

[2] http://ictclas.org/

lated into the other language using the Google translator, entity alignment is performed between relation mentions and their translations. Finally 4,747 Chinese relation mentions are successfully translated and aligned from English and vice versa, 4,936 English relation mentions are translated and aligned from Chinese.

SVMLIB (Chang and Lin, 2011) is selected as our classifier since it supports multi-class classification. The training parameters C (SVM) is set to 2.4 according to our previous work on relation extraction (Qian et al., 2010). Relation classification performance is evaluated using the standard Precision (P), Recall (R) and their harmonic average (F1) as well as *deficiency* measure (cf. latter in this section.). Overall performance scores are averaged over 10 runs. For each run, 1/40 and 1/5 randomly selected instances are used as the training and test set respectively while the remaining instances are used as the unlabeled set for further labeling during active learning iterations.

Methods for Comparison

For fair comparison, two baseline methods of supervised learning are included to augment their training sets with labeled instances during iterations. However, these labeled instances are chosen randomly from the corpus.

SL-MO (Supervised Learning with monolingual labeled instances): only the monolingual labeled instances are fed to the SVM classifiers for both Chinese and English relation classification respectively. The initial training data only contain L_c and L_e for Chinese and English respectively.

SL-CR (Supervised Learning with cross-lingual labeled instances): in addition to monolingual labeled instances (**SL-MO**), the training data for supervised learning contain labeled instances translated from the other language. That is, the initial training data contain L_c and L_{ct} for Chinese, or L_e and L_{et} for English. More important, at each iteration not only the labeled instances are added to the training data of its own language, but their translated instances are also added to the training data of the other language.

AL-MO (Active Learning with monolingual instances): labeled and unlabeled data for active learning only contain monolingual instances. No translated instances are involved. That is, the data contain L_c and U_c for Chinese, or L_e and U_e for English respectively. This is the normal active learning method applied to a single language.

AL-CR (Active Learning with cross-lingual instances): both the manually labeled instances and their translated ones are added to the respective training data. The initial training data contain L_c and L_{ct} for Chinese, or L_e and L_{et} for English. At each iteration, the n least confidently classified instances in U_c and U_e are labeled and added to the Chinese/English training data respectively. Their translated instances in U_{et} and U_{ct} are also added to the English/Chinese training data respectively.

AL-BI (Active Learning with bilingual labeled and unlabeled instances): similar to **AL-CR** with the exception that the unlabeled instances are chosen not by uncertainty scores in one language, but by the joint uncertainty scores in two languages. (cf. §4.4)

Evaluation Metric

Although learning curves are often used to evaluate the performance for active learning, it is preferable to quantitatively compare various active learning methods using a statistical metric *deficiency* (Schein and Ungar, 2007) defined as:

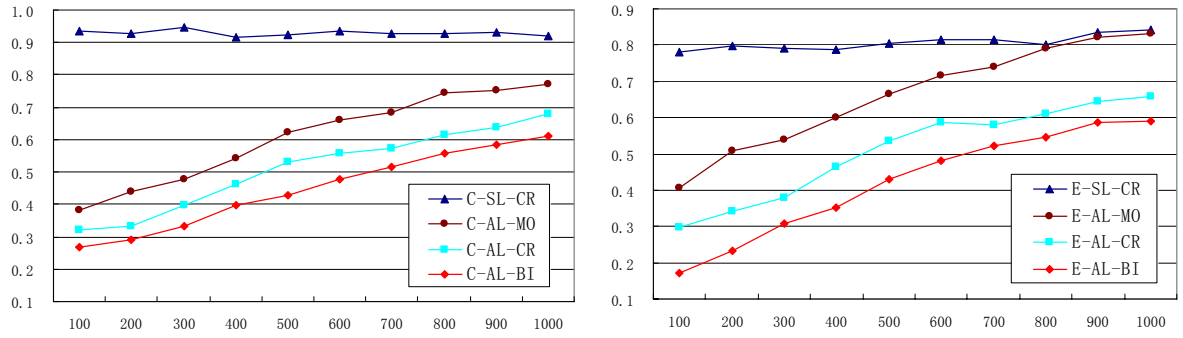
$$def_n(AL, REF) = \frac{\sum_{i=1}^{n}(F_n(REF) - F_i(AL))}{\sum_{i=1}^{n}(F_n(REF) - F_i(REF))} \quad (3)$$

Where n is the number of iterations involved in active learning and F_i is the F1-score of relation classification at the ith iteration. *REF* is the baseline active learning method and *AL* is an improved variant of *REF*, such as **AL-CR** or **AL-BI**. Essentially this *deficiency* metric measures the degree to which *REF* outperforms *AL*. Thus, smaller deficiency value (i.e. <1.0) indicates AL outperforms *REF* while a larger value (i.e. >1.0) indicates *AL* underperforms *REF*.

5.2 Experimental Results and Analysis

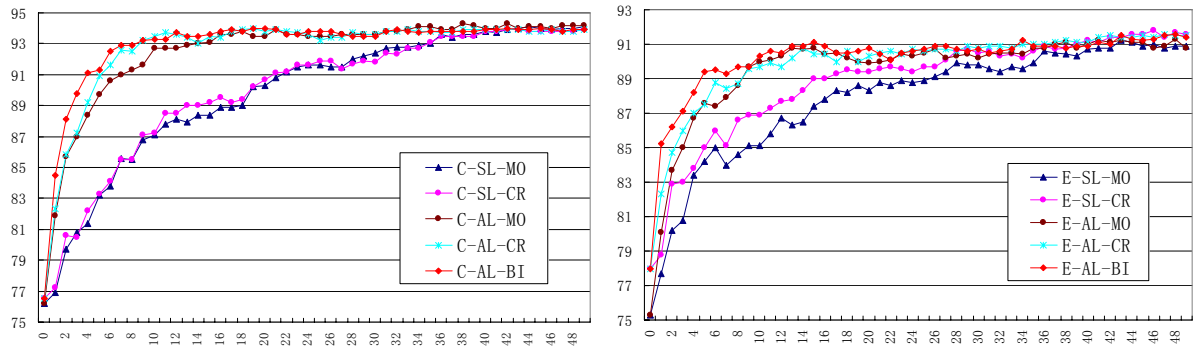
Comparison of overall deficiency

Table 1 compares the deficiency scores of relation classification on the Chinese (ACE2005c) and English corpora (ACE2005e) for various learning methods, i.e., **SL-CR**, **AL-MO**, **AL-CR** and **AL-BI**. Particularly, **SL-MO** is used as the baseline system against which deficiency scores for other methods are computed. The batch size n is set to 100 and iterations stop after all the unlabeled instances have run out of. Deficiency scores are averaged over 10 runs and the best ones are highlighted in bold font. Each run has a different test set and a different seed set.



(a) Chinese (b) English

Figure 5. Deficiency comparison for different batch sizes



(a) Chinese (b) English

Figure 6. Learning curves for different methods

The table shows that among the three active learning methods, bilingual active learning (**AL-BI**) achieves the best performance for both Chinese and English relation classification. This demonstrates that, bilingual active learning with jointly selecting the unlabeled instances can not only enhance relation classification for its own language, but also help relation classification for the other language due to the complementary nature of relation instances between Chinese and English.

Corpora	**SL-CR**	**AL-MO**	**AL-CR**	**AL-BI**
ACE2005c	0.934	0.383	0.323	**0.254**
ACE2005e	0.779	0.405	0.298	**0.160**

Table 1. Deficiency comparison of different methods

The table also shows the consistent utility of cross-lingual information for relation classification for both languages. When cross-lingual information is augmented, **SL-CR** outperforms **SL-MO** and **AL-CR** outperforms **AL-MO**.

Comparison of different batch sizes

Figure 5(a) and 5(b) illustrate the deficiency scores for four learning methods (**SL-CR**, **AL-MO**, **AL-CR** and **AL-BI**) against the **SL-MO** method with different batch sizes (n), where prefixes "C" and "E" denote Chinese and English respectively. The horizontal axes denote the range of n (<=1000) while the vertical ones denote the deficiency scores.

The figures show that the deficiency scores for three active learning methods run virtually parallel with each other while they increase monotonously with the batch size n. This suggests that for both Chinese and English **AL-BI** consistently performs best against other methods across a wide range of batch sizes, though the overall advantage of three active learning methods generally diminish.

Comparison of learning curves

In order to gain an intuition into how the performance evolves when the labeled instances are added into the training data during iterations, we depict the learning curves for various learning methods on the Chinese and English corpora in Fig. 6(a) and 6(b) respectively. The horizontal axes denote learning iterations while the vertical ones denote F1-scores. For simplicity of illustration the F1-scores are collected from one of the 10 runs.

The figures clearly demonstrate the performance difference for both languages among five methods at the beginning of iterations while F1-scores converge at the end of iterations. Particularly at the very outset, **AL-BI** outperforms other methods, quickly jumps to a very high point comparable to its best performance. However, after the 10th iteration the performance scores for the three AL variants tend to show trivial difference probably because most highly informative instances have already been added to the training data.

Comparison of annotation scale

In order to better compare BAL with other AL methods Figure 7 zooms out partial data on three AL methods in Fig. 6 and rescale the data for **AL-MO**, where "C" and "E" denote Chinese and English respectively. Likewise, the vertical axis denotes F1-scores while the horizontal axis denotes the number of instances labeled for **AL-CR** and **AL-BI**. However, for **AL-MO** that number is doubled. This figure tries to answer the question: to label *n* respective instances in both languages for BAL or to labeled *2n* instances in just one language for monolingual AL, can the former rival the latter?

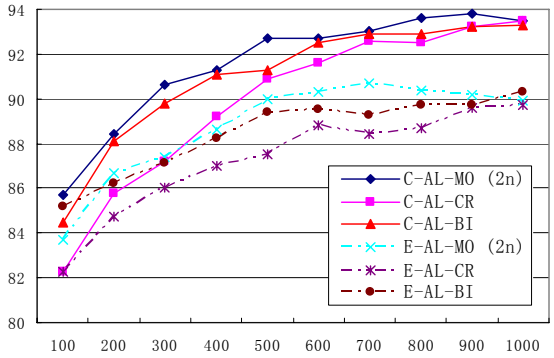


Figure 7. Comparison of annotation scale among three AL methods

The figure shows that for both Chinese and English, when the number of instances (*n*) to be labeled is no greater than 400, **AL-BI** with *n* instances can achieve comparable performance with **AL-MO** with *2n* instances. It implies that when the labeled instances are limited, labeling instances, half in one language and half in the other for BAL, is competitive against labeling the same total number of instances in just one language for monolingual AL, not to mention that the former can generate two relation extractors on two languages.

6 Conclusion

This paper proposes a bilingual active learning paradigm for Chinese and English relation classification. Given a small number of relation instances and a large number of unlabeled relation instances in both languages, we translate both the labeled and unlabeled instances in one language to the other as pseudo parallel corpora. After entity alignment, these labeled and unlabeled instances in both languages are fed into a bilingual active learning engine. Experiments with the task of relation classification on the ACE RDC 2005 Chinese and English corpora show that bilingual active learning can significantly outperforms monolingual active learning for both Chinese and English simultaneously. Moreover, we demonstrate that BAL across two languages can compete against monolingual AL when the annotation scale is limited, though the overall number of labeled instances remains the same.

For future work, on one hand, we plan to combine uncertainty sampling with diversity and informativeness measures; on the other hand, we intend to combine BAL with semi-supervised learning to further reduce human annotation efforts.

Acknowledgments

This research is supported by Grants 61373096, 61305088, 61273320, and 61331011 under the National Natural Science Foundation of China; Project 2012AA011102 under the "863" National High-Tech Research and Development of China; Grant 11KJA520003 under the Education Bureau of Jiangsu, China. We would like to thank the excellent and insightful comments from the three anonymous reviewers. Thanks also go to my colleague Dr. Shoushan Li for his helpful suggestions.

Reference

ACE. 2002-2007. Automatic Content Extraction. http://www.ldc.upenn.edu/Projects/ACE/

A. Brew, D. Greene, and P. Cunningham. 2010. Using crowdsourcing and active learning to track sentiment in online media. *ECAI'2010*: 145–150.

Y.S. Chan and D. Roth. 2011. Exploiting Syntactico-Semantic Structures for Relation Extraction. *ACL'2011*: 551-560

Y.S. Chan and H.T. Ng. 2007. Domain adaptation with active learning for word sense disambiguation. *ACL'2007*.

C.C. Chang and C.J. Lin. 2011. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(27):1-27.

W.X. Che, T. Liu, and S. Li. 2005. Automatic Extraction of Entity Relation (in Chinese). *Journal of Chinese Information Processing*, 19(2): 1-6.

J.X. Chen, D.H. Ji, and C. L. Tan. 2006. Relation Extraction using Label Propagation-based Semi-supervised Learning. *ACL/COLING'2006*: 129-136.

A. Culotta and J. Sorensen. 2004. Dependency tree kernels for relation extraction. *ACL'2004*: 423-439.

A. Culotta and A. McCallum. 2005. Reducing labeling effort for stuctured prediction tasks. *AAAI'2005*: 746–751.

S. P. Engelson and I. Dagan. 1996. Minimizing manual annotation cost in supervised training from corpora. ACL'1996: 319–326.

G. Haffari, M. Roy, and A. Sarkar. 2009. Active learning for statistical phrase-based machine translation. *NAACL'2009*: 415–423.

G. Haffari and A. Sarkar. 2009. Active learning for multilingual statistical machine translation. *ACL/IJCNLP'2009*: 181–189.

T. Hasegawa, S. Sekine, and R. Grishman. 2004. Discovering Relations among Named Entities from Large Corpora. *ACL'2004*.

Y.N. Hu, J.G. Shu, L.H. Qian, and Q.M. Qiao. 2013. Cross-lingual Relation Extraction based on Machine Translation (in Chinese). *Journal of Chinese Information Processing*, 27(5): 191-197.

R. Hwa. 2004. Sample selection for statistical parsing. *Computational Linguistics*, 30(3): 253–276.

S. Kim, M. Jeong, J. Lee, and G.G. Lee. 2010. A Cross-lingual Annotation Projection Approach for Relation Detection. *COLING'2010*: 564-571.

S. Kim and G.G. Lee. 2012. A Graph-based Cross-lingual Projection Approach for Weakly Supervised Relation Extraction. *ACL'2012*: 48-53.

S. Kim, Y. Song, K. Kim, J.W. Cha, and G.G. Lee. 2006. MMR-based active machine learning for bio named entity recognition. *HLT-NAACL'2006*: 69–72.

W.J. Li, P. Zhang, F.R. Wei, Y.X. Hou, and Q. Lu. 2008. A Novel Feature-based Approach to Chinese Entity Relation Extraction. *ACL'2008*: 89-92.

S.S. Li, S.F. Ju, G.D. Zhou, and X.J. Li. 2012. Active learning for imbalanced sentiment classification. *EMNLP-CoNLL'2012*: 139-148.

B. Lu, C.H. Tan, C. Cardie, and B.K. Tsou. 2011. Joint Bilingual Sentiment Classification with Unlabeled Parallel Corpora. *ACL'2011*: 320-330.

J. Oh, K. Uchimoto, and K. Torisawa. 2009. Bilingual Co-Training for Monolingual Hyponymy-Relation Acquisition. *ACL'2009*: 432-440.

M. Osborne and J. Baldridge. 2004. Ensemble based active learning for parse selection. *HLT-NAACL' 2004*: 89–96.

L.H. Qian, G.D. Zhou, F. Kong, and Q.M. Zhu. 2010. Clustering-based Stratified Seed Sampling for Semi-Supervised Relation Classification. *EMNLP2010*: 346-355.

L.H. Qian, G.D. Zhou, Q.M. Zhu, and P.D. Qian. 2008. Exploiting constituent dependencies for tree kernel-based semantic relation extraction. *COLING'2008*: 697-704.

R. Reichart, K. Tomanek, U. Hahn, and A. Rappoport. 2008. Multi-task active learning for linguistic annotations. *ACL'2008*: 861-869.

E. Ringger, P. McClanahan, R. Haertel, G. Busby, M. Carmen, J. Carroll, K. Seppi, and D. Lonsdale. 2007. Active learning for part-of-speech tagging: Accelerating corpus annotation. In *Proceedings of the Linguistic Annotation Workshop at ACL'2007*: 101–108.

T. Scheffer, C. Decomain, and S. Wrobel. 2001. Active hidden Markov models for information extraction. In Proceedings of the International Conference on Advances in Intelligent Data Analysis (CAIDA), pages 309–318.

A. I. Schein and L. H. Ungar. 2007. Active learning for logistic regression: an evaluation. *Machine Learning*, 68(3): 235-265.

P. Sebastian and M. Lapata. 2009. Cross-lingual annotation projection of semantic roles. *Journal of Artificial Intelligence Research*, 36(1): 307-340.

B. Settles and M. Craven. 2008. An Analysis of Active Learning Strategies for Sequence Labeling Tasks. *EMNLP'2008*: 1070–1079.

D. Shen, J. Zhang, J. Su, G.D. Zhou and C.-L. Tan. 2004. Multi-criteria-based active learning for named entity recognition. *ACL'2004*.

K. Tomanek and U. Hahn. 2009. Semi-Supervised Active Learning for Sequence Labeling. *ACL-IJCNLP'2009*: 1039-1047.

K. Tomanek, J. Wermter, and U. Hahn. 2007. An approach to text corpus construction which cuts annotation costs and maintains reusability of annotated data. *EMNLP-CoNLL'2007*: 486–495.

X.J. Wan. 2009. Co-Training for Cross-Lingual Sentiment Classification. *ACL-AFNLP'2009*: 235-243.

D. Yarowsky and G. Ngai. 2001. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. *NAACL'2001*: 1-8.

D. Yarowsky, G. Ngai, and R. Wicentorski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. *HLT'2001*:1-8.

H.H. Yu, L.H. Qian, G.D. Zhou, and Q.M. Zhu. 2010. Chinese Semantic Relation Extraction based on Unified Syntactic and Entity Semantic Tree (in Chinese). *Journal of Chinese Information Processing*, 24(5): 17-23.

D. Zelenko, C. Aone, and A. Richardella. 2003. Kernel Methods for Relation Extraction. *Journal of Machine Learning Research*, 3: 1083-1106.

Z. Zhang. 2004. Weakly-supervised relation classification for Information Extraction. *CIKM'2004*.

M. Zhang, J. Su, D. M. Wang, G. D. Zhou, and C. L. Tan. 2005. Discovering Relations between Named Entities from a Large Raw Corpus Using Tree Similarity-Based Clustering. *IJCNLP'2005*: 378-389.

M. Zhang, J. Zhang, J. Su, and G.D. Zhou. 2006. A Composite Kernel to Extract Relations between Entities with both Flat and Structured Features. *ACL/COLING'2006*: 825-832.

S.B. Zhao and R. Grishman. 2005. Extracting relations with integrated information using kernel methods. *ACL'2005*: 419-426.

G.D. Zhou, J.H. Li, L.H. Qian, and Q.M. Zhu. 2008. Semi-Supervised Learning for Relation Extraction. *IJCNLP'2008*: 32-38.

G.D. Zhou, J. Su, J. Zhang, and M. Zhang. 2005. Exploring various knowledge in relation extraction. *ACL'2005*: 427-434.

J.B. Zhu and E. Hovy. 2007. Active learning for word sense disambiguation with methods for addressing the class imbalance problem. *EMNLP-CoNLL'2007*: 783-790.

Learning Soft Linear Constraints with Application to Citation Field Extraction

Sam Anzaroot Alexandre Passos David Belanger Andrew McCallum
Department of Computer Science
University of Massachusetts, Amherst
{anzaroot, apassos, belanger, mccallum}@cs.umass.edu

Abstract

Accurately segmenting a citation string into fields for authors, titles, etc. is a challenging task because the output typically obeys various global constraints. Previous work has shown that modeling *soft constraints*, where the model is encouraged, but not require to obey the constraints, can substantially improve segmentation performance. On the other hand, for imposing *hard constraints*, dual decomposition is a popular technique for efficient prediction given existing algorithms for unconstrained inference. We extend dual decomposition to perform prediction subject to soft constraints. Moreover, with a technique for performing inference given soft constraints, it is easy to automatically generate large families of constraints and learn their costs with a simple convex optimization problem during training. This allows us to obtain substantial gains in accuracy on a new, challenging citation extraction dataset.

1 Introduction

Citation field extraction, an instance of information extraction, is the task of segmenting and labeling research paper citation strings into their constituent parts, including authors, editors, year, journal, volume, conference venue, etc. This task is important because citation data is often provided only in plain text; however, having an accurate structured database of bibliographic information is necessary for many scientometric tasks, such as mapping scientific sub-communities, discovering research trends, and analyzing networks of researchers. Automated citation field extraction needs further research because it has not yet reached a level of accuracy at which it can be practically deployed in real-world systems.

Hidden Markov models and linear-chain conditional random fields (CRFs) have previously been applied to citation extraction (Hetzner, 2008; Peng and McCallum, 2004) . These models support efficient dynamic-programming inference, but only model *local* dependencies in the output label sequence. However citations have strong *global* regularities not captured by these models. For example many book citations contain both an author section and an editor section, but none have two disjoint author sections. Since linear-chain models are unable to capture more than Markov dependencies, the models sometimes mislabel the *editor* as a second author. If we could enforce the global constraint that there should be only one author section, accuracy could be improved.

One framework for adding such global constraints into tractable models is *constrained inference*, in which at inference time the original model is augmented with restrictions on the outputs such that they obey certain global regularities. When hard constraints can be encoded as linear equations on the output variables, and the underlying model's inference task can be posed as linear optimization, one can formulate this constrained inference problem as an *integer linear program* (ILP) (Roth and Yih, 2004). Alternatively, one can employ *dual decomposition* (Rush et al., 2010). Dual decompositions's advantage over ILP is is that it can leverage existing inference algorithms for the original model as a black box. Such a modular algorithm is easy to implement, and works quite well in practice, providing certificates of optimality for most examples.

The above two approaches have previously been applied to impose *hard* constraints on a model's output. On the other hand, recent work has demonstrated improvements in citation field extraction by imposing *soft* constraints (Chang et al., 2012). Here, the model is not required obey the global constraints, but merely pays a penalty for their vi-

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 593–602,
Baltimore, Maryland, USA, June 23-25 2014.

4 ·ref-marker [J. first D. middle Monk ,last person]authors [Cardinal Functions on Boolean Algebra ,]title [Lectures in Mathematics , ETH Zurich , series Birkhause Verlag , publisher Basel , Boston , Berlin , address 1990 . year date]venue

Figure 1: Example labeled citation

olation.

This paper introduces a novel method for imposing soft constraints via dual decomposition. We also propose a method for learning the penalties the prediction problem incurs for violating these soft constraints. Because our learning method drives many penalties to zero, it allows practitioners to perform 'constraint selection,' in which a large number of automatically-generated candidate global constraints can be considered and automatically culled to a smaller set of useful constraints, which can be run quickly at test time.

Using our new method, we are able to incorporate not only all the soft global constraints of Chang et al. (2012), but also far more complex data-driven constraints, while also providing stronger optimality certificates than their beam search technique. On a new, more broadly representative, and challenging citation field extraction data set, we show that our methods achieve a 17.9% reduction in error versus a linear-chain conditional random field. Furthermore, we demonstrate that our inference technique can use and benefit from the constraints of Chang et al. (2012), but that including our data-driven constraints on top of these is beneficial. While this paper focusses on an application to citation field extraction, the novel methods introduced here would easily generalize to many problems with global output regularities.

2 Background

2.1 Structured Linear Models

The overall modeling technique we employ is to add soft constraints to a simple model for which we have an existing efficient prediction algorithm. For this underlying model, we employ a chain-structured conditional random field (CRF), since CRFs have been shown to perform better than other simple unconstrained models like hidden markov models for citation extraction (Peng and McCallum, 2004). We produce a prediction by performing *MAP inference* (Koller and Friedman, 2009).

The MAP inference task in a CRF be can expressed as an optimization problem with a linear objective (Sontag, 2010; Sontag et al., 2011). Here, we define a binary indicator variable for each candidate setting of each factor in the graphical model. Each of these indicator variables is associated with the score that the factor takes on when it has the indictor variable's corresponding value. Since the log probability of some y in the CRF is proportional to sum of the scores of all the factors, we can concatenate the indicator variables as a vector y and the scores as a vector w and write the MAP problem as

$$\begin{aligned} \textbf{max.} \quad & \langle w, y \rangle \\ \textbf{s.t.} \quad & y \in \mathcal{U}, \end{aligned} \tag{1}$$

where the set $\mathcal{U}$ represents the set of valid configurations of the indicator variables. Here, the constraints are that all neighboring factors agree on the components of y in their overlap.

Structured Linear Models are the general family of models where prediction requires solving a problem of the form (1), and they do not always correspond to a probabilistic model. The algorithms we present in later sections for handling soft global constraints and for learning the penalties of these constraints can be applied to general structured linear models, not just CRFs, provided we have an available algorithm for performing MAP inference.

2.2 Dual Decomposition for Global Constraints

In order to perform prediction subject to various global constraints, we may need to augment the problem (1) with additional constraints. Dual Decomposition is a popular method for performing MAP inference in this scenario, since it leverages known algorithms for MAP in the base problem where these extra constraints have not been added (Komodakis et al., 2007; Sontag et al., 2011; Rush and Collins, 2012). In this case, the MAP problem can be formulated as a structured linear model similar to equation (1), for which we have a MAP algorithm, but where we have imposed some additional constraints $Ay \leq b$ that no longer allow us to use the algorithm. In other

Algorithm 1 DD: projected subgradient for dual decomposition with hard constraints

1: **while** has not converged **do**
2: $y^{(t)} = \text{argmax}_{y \in \mathcal{U}} \langle w + A^T \lambda, y \rangle$
3: $\lambda^{(t)} = \Pi_{0 \leq \cdot} \left[\lambda^{(t-1)} - \eta^{(t)} (Ay - b) \right]$

words, we consider the problem

$$\begin{aligned} \textbf{max.} \quad & \langle w, y \rangle \\ \textbf{s.t.} \quad & y \in \mathcal{U} \\ & Ay \leq b, \end{aligned} \tag{2}$$

for an arbitrary matrix A and vector b. We can write the Lagrangian of this problem as

$$L(y, \lambda) = \langle w, y \rangle + \lambda^T (Ay - b). \tag{3}$$

Regrouping terms and maximizing over the primal variables, we have the dual problem

$$\textbf{min.}_\lambda D(\lambda) = \max_{y \in \mathcal{U}} \langle w + A^T \lambda, y \rangle - \lambda^T b. \tag{4}$$

For any λ, we can evaluate the dual objective $D(\lambda)$, since the maximization in (4) is of the same form as the original problem (1), and we assumed we had a method for performing MAP in this. Furthermore, a subgradient of $D(\lambda)$ is $Ay^* - b$, for an y^* which maximizes this inner optimization problem. Therefore, we can minimize $D(\lambda)$ with the projected subgradient method (Boyd and Vandenberghe, 2004), and the optimal y can be obtained when evaluating $D(\lambda^*)$. Note that the subgradient of $D(\lambda)$ is the amount by which each constraint is violated by λ when maximizing over y.

Algorithm 1 depicts the basic projected subgradient descent algorithm for dual decomposition. The projection operator Π consists of truncating all negative coordinates of λ to 0. This is necessary because λ is a vector of dual variables for inequality constraints. The algorithm has converged when each constraint is either satisfied by $y^{(t)}$ with equality or its corresponding component of λ is 0, due to complimentary slackness (Boyd and Vandenberghe, 2004).

3 Soft Constraints in Dual Decomposition

We now introduce an extension of Algorithm 1 to handle soft constraints. In our formulation, a soft-constrained model imposes a penalty for each unsatisfied constraint, proportional to the amount by which it is violated. Therefore, our derivation parallels how soft-margin SVMs are derived from hard-margin SVMs by introducing auxiliary slack variables (Cortes and Vapnik, 1995). Note that when performing MAP subject to soft constraints, optimal solutions might not satisfy some constraints, since doing so would reduce the model's score by too much.

Consider the optimization problems of the form:

$$\begin{aligned} \textbf{max.} \quad & \langle w, y \rangle - \langle c, z \rangle \\ \textbf{s.t.} \quad & y \in \mathcal{U} \\ & Ay - b \leq z \\ & -z \leq 0, \end{aligned} \tag{5}$$

For positive c_i, it is clear that an optimal z_i will be equal to the degree to which $a_i^T y \leq b_i$ is violated. Therefore, we pay a cost c_i times the degree to which the ith constraint is violated, which mirrors how slack variables are used to represent the hinge loss for SVMs. Note that c_i has to be positive, otherwise this linear program is unbounded and an optimal value can be obtained by setting z_i to infinity.

Using a similar construction as in section 2.2 we write the Lagrangian as:

$$\begin{aligned} L(y, z, \lambda, \mu) = & \langle w, y \rangle - \langle c, z \rangle \\ & + \lambda^T (Ay - b - z) + \mu^T (-z). \end{aligned} \tag{6}$$

The optimality constraints with respect to z tell us that $-c - \lambda - \mu = 0$, hence $\mu = -c - \lambda$. Substituting, we have

$$L(y, \lambda) = \langle w, y \rangle + \lambda^T (Ay - b), \tag{7}$$

except the constraint that $\mu = -c - \lambda$ implies that for μ to be positive $\lambda \leq c$.

Since this Lagrangian has the same form as equation (3), we can also derive a dual problem, which is the same as in equation (4), with the additional constraint that each λ_i can not be bigger than its cost c_i. In other words, the dual problem can not penalize the violation of a constraint more than the soft constraint model in the primal would penalize you if you violated it.

This optimization problem can still be solved with projected subgradient descent and is depicted in Algorithm 2. The only modifications to Algorithm 1 are replacing the coordinate-wise projection $\Pi_{0 \leq \cdot}$ with $\Pi_{0 \leq \cdot \leq c}$ and how we check for convergence. Now, we check for the KKT conditions of (5), where for every constraint i, either the constraint is satisfied with equality, $\lambda_i = 0$, or $\lambda_i = c_i$.

Algorithm 2 Soft-DD: projected subgradient for dual decomposition with soft constraints

1: **while** has not converged **do**
2: $y^{(t)} = \text{argmax}_{y \in \mathcal{U}} \langle w + A^T \lambda, y \rangle$
3: $\lambda^{(t)} = \Pi_{0 \leq \cdot \leq c} \left[\lambda^{(t-1)} - \eta^{(t)} (Ay - b) \right]$

Therefore, implementing soft-constrained dual decomposition is as easy as implementing hard-constrained dual decomposition, and the per-iteration complexity is the same. We encourage further applications of soft-constraint dual decomposition to existing and new NLP problems.

3.1 Learning Penalties

One consideration when using soft v.s. hard constraints is that soft constraints present a new training problem, since we need to choose the vector c, the penalties for violating the constraints. An important property of problem (5) in the previous section is that it corresponds to a structured linear model over y and z. Therefore, we can apply known training algorithms for estimating the parameters of structured linear models to choose c.

All we need to employ the structured perceptron algorithm (Collins, 2002) or the structured SVM algorithm (Tsochantaridis et al., 2004) is a black-box procedure for performing MAP inference in the structured linear model given an arbitrary cost vector. Fortunately, the MAP problem for (5) can be solved using Soft-DD, in Algorithm 2.

Each penalty c_i has to be non-negative; otherwise, the optimization problem in equation (5) is ill-defined. This can be ensured by simple modifications of the perceptron and subgradient descent optimization of the structured SVM objective simply by truncating c coordinate-wise to be non-negative at every learning iteration.

Intuitively, the perceptron update increases the penalty for a constraint if it is satisfied in the ground truth and not in an inferred prediction, and decreases the penalty if the constraint is satisfied in the prediction and not the ground truth. Since we truncate penalties at 0, this suggests that we will learn a penalty of 0 for constraints in three categories: constraints that do not hold in the ground truth, constraints that hold in the ground truth but are satisfied in practice by performing inference in the base CRF model, and constraints that are satisfied in practice as a side-effect of imposing non-zero penalties on some other constraints . A similar analysis holds for the structured SVM approach.

Therefore, we can view learning the values of the penalties not just as parameter tuning, but as a means to perform 'constraint selection,' since constraints that have a penalty of 0 can be ignored. This property allows us to consider large families of constraints, from which the useful ones are automatically identified.

We found it beneficial, though it is not theoretically necessary, to learn the constraints on a held-out development set, separately from the other model parameters, as during training most constraints are satisfied due to overfitting, which leads to an underestimation of the relevant penalties.

4 Citation Extraction Data

We consider the UMass citation dataset, first introduced in Anzaroot and McCallum (2013). It has over 1800 citation from many academic fields, extracted from the arXiv. This dataset contains both coarse-grained and fine-grained labels; for example it contains labels for the segment of all authors, segments for each individual author, and for the first and last name of each author. There are 660 citations in the development set and 367 citation in the test set.

The labels in the UMass dataset are a concatenation of labels from a hierarchically-defined schema. For example, a first name of an author is tagged as: *authors/person/first*. In addition, individual tokens are labeled using a BIO label schema for each level in the hierarchy. BIO is a commonly used labeling schema for information extraction tasks. BIO labeling allows individual labels on tokens to label segmentation information as well as labels for the segments. In this schema, labels that begin segments are prepended with a *B*, labels that continue a segment are prepended with an *I*, and tokens that don't have a labeling in this schema are given an *O* label. For example, in a hierarchical BIO label schema the first token in the first name for the second author may be labeled as: *I-authors/B-person/B-first*.

An example labeled citation in this dataset can be viewed in figure 1.

5 Global Constraints for Citation Extraction

5.1 Constraint Templates

We now describe the families of global constraints we consider for citation extraction. Note these

constraints are all linear, since they depend only on the counts of each possible conditional random field label. Moreover, since our labels are BIO-encoded, it is possible, by counting B tags, to count how often each citation tag itself appears in a sentence. The first two families of constraints that we describe are general to any sequence labeling task while the last is specific to hierarchical labeling such as available in the UMass dataset.

Our sequence output is denoted as y and an element of this sequence is y_k.

We denote $[[y_k = i]]$ as the function that outputs 1 if y_k has a 1 at index i and 0 otherwise. Here, y_k represents an output tag of the CRF, so if $[[y_k = i]]$ = 1, then we have that y_k was given a label with index i.

5.2 Singleton Constraints

Singleton constraints ensure that each label can appear at most once in a citation. These are same global constraints that were used for citation field extraction in Chang et al. (2012). We define $s(i)$ to be the number of times the label with index i is predicted in a citation, formally:

$$s(i) = \sum_{y_k \in y} [[y_k = i]]$$

The constraint that each label can appear at most once takes the form:

$$s(i) <= 1$$

5.3 Pairwise Constraints

Pairwise constraints are constraints on the counts of two labels in a citation. We define $z_1(i, j)$ to be

$$z_1(i, j) = \sum_{y_k \in y} [[y_k = i]] + \sum_{y_k \in y} [[y_k = j]]$$

and $z_2(i, j)$ to be

$$z_2(i, j) = \sum_{y_k \in y} [[y_k = i]] - \sum_{y_k \in y} [[y_k = j]]$$

We consider all constraints of the forms: $z(i, j) \leq 0, 1, 2, 3$ and $z(i, j) \geq 0, 1, 2, 3$.

Note that some pairs of these constraints are redundant or logically incompatible. However, we are using them as soft constraints, so these constraints will not necessarily be satisfied by the output of the model, which eliminates concern over enforcing logically impossible outputs. Furthermore, in section 3.1 we described how our procedure for learning penalties will drive some penalties to 0, which effectively removes them from our set of constraints we consider. It can be shown, for example, that we will never learn non-zero penalties for certain pairs of logically incompatible constraints using the perceptron-style algorithm described in section 3.1 .

5.4 Hierarchical Equality Constraints

The labels in the citation dataset are hierarchical labels. This means that the labels are the concatenation of all the levels in the hierarchy. We can create constraints that are dependent on only one or couple of elements in the hierarchy.

We define $C(x, i)$ as the function that returns 1 if the output x contains the label i in the hierarchy and 0 otherwise. We define $e(i, j)$ to be

$$e(i, j) = \sum_{y_k \in y} [[C(y_k, i)]] - \sum_{y_k \in y} [[C(y_k, j)]]$$

Hierarchical equality constraints take the forms:

$$e(i, j) \geq 0 \quad (8)$$
$$e(i, j) \leq 0 \quad (9)$$

5.5 Local constraints

We constrain the output labeling of the chain-structured CRF to be a valid BIO encoding. This both improves performance of the underlying model when used without global constraints, as well as ensures the validity of the global constraints we impose, since they operate only on *B* labels. The constraint that the labeling is valid BIO can be expressed as a collection of pairwise constraints on adjacent labels in the sequence. Rather than enforcing these constraints using dual decomposition, they can be enforced directly when performing MAP inference in the CRF by modifying the dynamic program of the Viterbi algorithm to only allow valid pairs of adjacent labels.

5.6 Constraint Pruning

While the techniques from section 3.1 can easily cope with a large numbers of constraints at training time, this can be computationally costly, specially if one is considering very large constraint families. This is problematic because the size

Unconstrained	[17]ref-marker [D.first Sivia ,last person J.first Skilling ,last person]authors [Data Analysis : A Bayesian Tutorial ,booktitle Oxford University Press , publisher 2006 year date]venue
Constrained	[17]ref-marker [D.first Sivia ,last person J.first Skilling ,last person]authors Data Analysis : A Bayesian Tutorial ,title [Oxford University Press , publisher 2006 year date]venue
Unconstrained	[Sobol' ,last I.first M.middle person]authors [(1990) .year]date [On sensitivity estimation for nonlinear mathematical models .]title [Matematicheskoe Modelirovanie ,journal 2volume (1) :number 112–118 .pages (In Russian) . status]venue
Constrained	[Sobol' ,last I.first M.middle person]authors [(1990) .year]date [On sensitivity estimation for nonlinear mathematical models .]title [Matematicheskoe Modelirovanie ,journal 2volume (1) :number 112–118 .pages (In Russian) . language]venue

Figure 2: Two examples where imposing soft global constraints improves field extraction errors. Soft-DD converged in 1 iteration on the first example, and 7 iterations on the second. When a reference is citing a book and not a section of the book, the correct labeling of the name of the book is title. In the first example, the baseline CRF incorrectly outputs booktitle, but this is fixed by Soft-DD, which penalizes outputs based on the constraint that booktitle should co-occur with an address label. In the second example, the unconstrained CRF output violates the constraint that title and status labels should not co-occur. The ground truth labeling also violates a constraint that title and language labels should not co-occur. At convergence of the Soft-DD algorithm, the correct labeling of language is predicted, which is possible because of the use of soft constraints.

Constraints	F1 score	Sparsity	# of cons
Baseline	94.44		
Only-one	94.62	0%	3
Hierarchical	94.55	56.25%	16
Pairwise	95.23	43.19%	609
All	**95.39**	32.96%	628
All DD	94.60	0%	628

Table 1: Set of constraints learned and F1 scores. The last row depicts the result of inference using all constraints as hard constraints.

of some constraint families we consider grows quadratically with the number of candidate labels, and there are about 100 in the UMass dataset. Such a family consists of constraints that the sum of the counts of two different label types has to be bounded (a useful example is that there can't be more than one out of "phd thesis" and "journal"). Therefore, quickly pruning bad constraints can save a substantial amount of training time, and can lead to better generalization.

To do so, we calculate a score that estimates how useful each constraint is expected to be. Our score compares how often the constraint is violated in the ground truth examples versus our predictions. Here, prediction is done with respect to the base chain-structured CRF tagger and does not include global constraints. Note that it may make sense to consider a constraint that is sometimes violated in the ground truth, as the penalty learning algorithm can learn a small penalty for it, which will allow it to be violated some of the time. Our importance score is defined as, for each constraint c on labeled set D,

$$imp(c) = \frac{\sum_{d \in D} [[max_y w_d^T y]]_c}{\sum_{d \in D} [[y_d]]_c}, \qquad (10)$$

where $[[y]]_c$ is 1 if the constraint is violated on output y and 0 otherwise. Here, y_d denotes the ground truth labeling and w_d is the vector of scores for the CRF tagger.

We prune constraints by picking a cutoff value for $imp(c)$. A value of $imp(c)$ above 1 implies that the constraint is more violated on the predicted examples than on the ground truth, and hence that we might want to keep it.

We also find that the constraints that have the largest imp values are semantically interesting.

6 Related Work

There are multiple previous examples of augmenting chain-structured sequence models with terms capturing global relationships by expanding the chain to a more complex graphical model with non-local dependencies between the outputs. Inference in these models can be performed, for example, with loopy belief propagation (Bunescu and Mooney, 2004; Sutton and McCallum, 2004) or Gibbs sampling (Finkel et al., 2005). Belief propagation is prohibitively expensive in our

model due to the high cardinalities of the output variables and of the global factors, which involve all output variables simultaneously. There are various methods for exploiting the combinatorial structure of these factors, but performance would still have higher complexity than our method. While Gibbs sampling has been shown to work well tasks such as named entity recognition (Finkel et al., 2005), our previous experiments show that it does not work well for citation extraction, where it found only low-quality solutions in practice because the sampling did not mix well, even on a simple chain-structured CRF.

Recently, dual decomposition has become a popular method for solving complex structured prediction problems in NLP (Koo et al., 2010; Rush et al., 2010; Rush and Collins, 2012; Paul and Eisner, 2012; Chieu and Teow, 2012). Soft constraints can be implemented inefficiently using hard constraints and dual decomposition— by introducing copies of output variables and an auxiliary graphical model, as in Rush et al. (2012). However, at every iteration of dual decomposition, MAP must be run in this auxiliary model. Furthermore the copying of variables doubles the number of iterations needed for information to flow between output variables, and thus slows convergence. On the other hand, our approach to soft constraints has identical per-iteration complexity as for hard constraints, and is a very easy modification to existing hard constraint code.

Initial work in machine learning for citation extraction used Markov models with no global constraints. Hidden Markov models (HMMs), were originally employed for automatically extracting information from research papers on the CORA dataset (Seymore et al., 1999; Hetzner, 2008). Later, CRFs were shown to perform better on CORA, improving the results from the Hmm's token-level F1 of 86.6 to 91.5 with a CRF(Peng and McCallum, 2004).

Recent work on globally-constrained inference in citation extraction used an HMM^{CCM}, which is an HMM with the addition of global features that are restricted to have positive weights (Chang et al., 2012). Approximate inference is performed using beam search. This method increased the HMM token-level accuracy from 86.69 to 93.92 on a test set of 100 citations from the CORA dataset. The global constraints added into the model are simply that each label only occurs once per citation. This approach is limited in its use of an HMM as an underlying model, as it has been shown that CRFs perform significantly better, achieving 95.37 token-level accuracy on CORA (Peng and McCallum, 2004). In our experiments, we demonstrate that the specific global constraints used by Chang et al. (2012) help on the UMass dataset as well.

7 Experimental Results

Our baseline is the one used in Anzaroot and McCallum (2013), with some labeling errors removed. This is a chain-structured CRF trained to maximize the conditional likelihood using L-BFGS with L2 regularization.

We use the same features as Anzaroot and McCallum (2013), which include word type, capitalization, binned location in citation, regular expression matches, and matches into lexicons. In addition, we use a rule-based segmenter that segments the citation string based on punctuation as well as probable start or end segment words (e.g. 'in' and 'volume'). We add a binary feature to tokens that correspond to the start of a segment in the output of this simple segmenter. This final feature improves the F1 score on the cleaned test set from 94.0 F1 to 94.44 F1, which we use as a baseline score.

We then use the development set to learn the penalties for the soft constraints, using the perceptron algorithm described in section 3.1. MAP inference in the model with soft constraints is performed using Soft-DD, shown in Algorithm 2.

We instantiate constraints from each template in section 5.1, iterating over all possible labels that contain a *B* prefix at any level in the hierarchy and pruning all constraints with $imp(c) < 2.75$ calculated on the development set. We asses performance in terms of field-level F1 score, which is the harmonic mean of precision and recall for predicted segments.

Table 1 shows how each type of constraint family improved the F1 score on the dataset. Learning all the constraints jointly provides the largest improvement in F1 at 95.39. This improvement in F1 over the baseline CRF as well as the improvement in F1 over using only-one constraints was shown to be statistically significant using the Wilcoxon signed rank test with p-values < 0.05. In the all-constraints settings, 32.96% of the constraints have a learned parameter of 0, and therefore only

Stop	F1 score	Convergence	Avg Iterations
1	94.44	76.29%	1.0
2	95.07	83.38%	1.24
5	95.12	95.91%	1.61
10	95.39	99.18%	1.73

Table 2: Performance from terminating Soft-DD early. Column 1 is the number of iterations we allow each example. Column 3 is the % of test examples that converged. Column 4 is the average number of necessary iterations, a surrogate for the slowdown over performing unconstrained inference.

421 constraints are active. Soft-DD converges, and thus solves the constrained inference problem exactly, for all test set examples after at most 41 iterations. Running Soft-DD to convergence requires 1.83 iterations on average per example. Since performing inference in the CRF is by far the most computationally intensive step in the iterative algorithm, this means our procedure requires approximately twice as much work as running the baseline CRF on the dataset. On examples where unconstrained inference does not satisfy the constraints, Soft-DD converges after 4.52 iterations on average. For 11.99% of the examples, the Soft-DD algorithm satisfies constraints that were not satisfied during unconstrained inference, while in the remaining 11.72% Soft-DD converges with some constraints left unsatisfied, which is possible since we are imposing them as soft constraints.

We could have enforced these constraints as hard constraints rather than soft ones. This experiment is shown in the last row of Table 1, where F1 only improves to 94.6. In addition, running the DD algorithm with these constraints takes 5.21 iterations on average per example, which is 2.8 times slower than Soft-DD with learned penalties.

In Figure 2, we analyze the performance of Soft-DD when we don't necessarily run it to convergence, but stop after a fixed number of iterations on each test set example. We find that a large portion of our gain in accuracy can be obtained when we allow ourselves as few as 2 dual decomposition iterations. However, this only amounts to 1.24 times as much work as running the baseline CRF on the dataset, since the constraints are satisfied immediately for many examples.

In Figure 2 we consider two applications of our Soft-DD algorithm, and provide analysis in the caption.

We train and evaluate on the UMass dataset instead of CORA, because it is significantly larger, has a useful finer-grained labeling schema, and its annotation is more consistent. We were able to obtain better performance on CORA using our baseline CRF than the HMM^{CCM} results presented in Chang et al. (2012), which include soft constraints. Given this high performance of our base model on CORA, we did not apply our Soft-DD algorithm to the dataset. Furthermore, since the dataset is so small, learning the penalties for our large collection of constraints is difficult, and test set results are unreliable. Rather than compare our work to Chang et al. (2012) via results on CORA, we apply their constraints on the UMass data using Soft-DD and demonstrate accuracy gains, as discussed above.

7.1 Examples of learned constraints

We now describe a number of the useful constraints that receive non-zero learned penalties and have high importance scores, defined in Section 5.6. The importance score of a constraint provides information about how often it is violated by the CRF, but holds in the ground truth, and a non-zero penalty implies we enforce it as a soft constraint at test time.

The two singleton constraints with highest importance score are that there should only be at most one title segment in a citation and that there should be at most one author segment in a citation. The only one author constraint is particularly useful for correctly labeling editor segments in cases where unconstrained inference mislabels them as author segments. As can be seen in Table 3, editor fields are among the most improved with our new method, largely due to this constraint.

The two hierarchical constraints with the highest importance scores with non-zero learned penalties constrain the output such that number of person segments does not exceed the number of first segments and vice-versa. Together, these constraints penalize outputs in which the number of person segments do not equal the number of first segments, i.e., every author should have a first name.

One important pairwise constraint penalizes outputs in which thesis segments don't co-occur with school segments. School segments label the name of the university that the thesis was submitted to. The application of this constraint increases the performance of the model on school segments

Label	U	C	+
venue/series	35.29	66.67	31.37
venue/editor/person/first	66.67	94.74	28.07
venue/school	40.00	66.67	26.67
venue/editor/person/last	75.00	94.74	19.74
venue/editor	77.78	90.00	12.22
venue/editor/person/middle	81.82	91.67	9.85

Table 3: Labels with highest improvement in F1. U is in unconstrained inference. C is the results of constrained inference. + is the improvement in F1.

dramatically, as can be seen in table 3.

An interesting form of pairwise constraints penalize outputs in which some labels do not co-occur with other labels. Some examples of constraints in this form enforce that journal segments should co-occur with pages segments and that booktitle segments should co-occur with address segments. An example of the latter constraint being employed during inference is the first example in Figure 2. Here, the constrained inference penalizes output which contains a booktitle segment but no address segment. This penalization leads allows the constrained inference to correctly label the booktitle segment as a title segment.

The above example constraints are almost always satisfied on the ground truth, and would be useful to enforce as hard constraints. However, there are a number of learned constraints that are often violated on the ground truth but are still useful as soft constraints. Take, for example, the constraint that the number of number segments does not exceed the number of booktitle segments, as well as the constraint that it does not exceed the number of journal segments. These constraints are moderately violated on ground truth examples, however. For example, when booktitle segments co-occur with number segments but not with journal segments, the second constraint is violated. It is still useful to impose these soft constraints, as strong evidence from the CRF allows us to violate them, and they can guide the model to good predictions when the CRF is unconfident.

8 Conclusion

We introduce a novel modification to the standard projected subgradient dual decomposition algorithm for performing MAP inference subject to hard constraints to one for performing MAP in the presence of soft constraints. In addition, we offer an easy-to-implement procedure for learning the penalties on soft constraints. This method drives many penalties to zero, which allows users to automatically discover discriminative constraints from large families of candidates.

We show via experiments on a recent substantial dataset that using soft constraints, and selecting which constraints to use with our penalty-learning procedure, can lead to significant gains in accuracy. We achieve a 17% gain in accuracy over a chain-structured CRF model, while only needing to run MAP in the CRF an average of less than 2 times per example. This minor incremental cost over Viterbi, plus the fact that we obtain certificates of optimality on 100% of our test examples in practice, suggests the usefulness of our algorithm for large-scale applications. We encourage further use of our Soft-DD procedure for other structured prediction problems.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval, in part by DARPA under agreement number FA8750-13-2-0020, in part by NSF grant #CNS-0958392, and in part by IARPA via DoI/NBC contract #D11PC20152. The U.S. Government is authorized to reproduce and distribute reprint for Governmental purposes notwithstanding any copyright annotation thereon. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

Sam Anzaroot and Andrew McCallum. 2013. A new dataset for fine-grained citation field extraction. In *ICML Workshop on Peer Reviewing and Publishing Models*.

Stephen Poythress Boyd and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.

Razvan Bunescu and Raymond J Mooney. 2004. Collective information extraction with relational markov networks. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 438. Association for Computational Linguistics.

M. Chang, L. Ratinov, and D. Roth. 2012. Structured learning with constrained conditional models. *Machine Learning*, 88(3):399–431, 6.

Hai Leong Chieu and Loo-Nin Teow. 2012. Combining local and non-local information with dual decomposition for named entity recognition from text.

In *Information Fusion (FUSION), 2012 15th International Conference on*, pages 231–238. IEEE.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.

Erik Hetzner. 2008. A simple method for citation metadata extraction using hidden markov models. In *Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries*, pages 280–284. ACM.

Daphne Koller and Nir Friedman. 2009. *Probabilistic graphical models: principles and techniques*. The MIT Press.

Nikos Komodakis, Nikos Paragios, and Georgios Tziritas. 2007. Mrf optimization via dual decomposition: Message-passing revisited. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE.

Terry Koo, Alexander M Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1288–1298. Association for Computational Linguistics.

Michael J Paul and Jason Eisner. 2012. Implicitly intersecting weighted automata using dual decomposition. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 232–242. Association for Computational Linguistics.

Fuchun Peng and Andrew McCallum. 2004. Accurate information extraction from research papers using conditional random fields. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 329–336, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Dan Roth and Wen-tau Yih. 2004. *A linear programming formulation for global inference in natural language tasks*. Defense Technical Information Center.

Alexander M. Rush and Michael Collins. 2012. A tutorial on dual decomposition and lagrangian relaxation for inference in natural language processing. *J. Artif. Intell. Res. (JAIR)*, 45:305–362.

Alexander M Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1–11. Association for Computational Linguistics.

Alexander M Rush, Roi Reichart, Michael Collins, and Amir Globerson. 2012. Improved parsing and pos tagging using inter-sentence consistency constraints. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1434–1444.

Kristie Seymore, Andrew McCallum, Roni Rosenfeld, et al. 1999. Learning hidden markov model structure for information extraction. In *AAAI-99 Workshop on Machine Learning for Information Extraction*, pages 37–42.

David Sontag, Amir Globerson, and Tommi Jaakkola. 2011. Introduction to dual decomposition for inference. In Suvrit Sra, Sebastian Nowozin, and Stephen J. Wright, editors, *Optimization for Machine Learning*. MIT Press.

David Sontag. 2010. *Approximate Inference in Graphical Models using LP Relaxations*. Ph.D. thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science.

Charles Sutton and Andrew McCallum. 2004. Collective segmentation and labeling of distant entities in information extraction. Technical report, DTIC Document.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the twenty-first international conference on Machine learning*, page 104. ACM.

A Study of Concept-based Weighting Regularization for Medical Records Search

Yue Wang, Xitong Liu, Hui Fang
Department of Electrical & Computer Engineering,
University of Delaware, USA
{wangyue,xtliu,hfang}@udel.edu

Abstract

An important search task in the biomedical domain is to find medical records of patients who are qualified for a clinical trial. One commonly used approach is to apply NLP tools to map terms from queries and documents to concepts and then compute the relevance scores based on the concept-based representation. However, the mapping results are not perfect, and none of previous work studied how to deal with them in the retrieval process. In this paper, we focus on addressing the limitations caused by the imperfect mapping results and study how to further improve the retrieval performance of the concept-based ranking methods. In particular, we apply axiomatic approaches and propose two weighting regularization methods that adjust the weighting based on the relations among the concepts. Experimental results show that the proposed methods are effective to improve the retrieval performance, and their performances are comparable to other top-performing systems in the TREC Medical Records Track.

1 Introduction

With the increasing use of electronic health records, it becomes urgent to leverage this rich information resource about patients' health conditions to transform research in health and medicine. As an example, when developing a cohort for a clinical trial, researchers need to identify patients matching a set of clinical criteria based on their medical records during their hospital visits (Safran et al., 2007; Friedman et al., 2010). This selection process is clearly a domain-specific retrieval problem, which searches for *relevant medical records* that contain useful information about their corresponding patients' qualification to the criteria specified in a query, e.g., "female patient with breast cancer with mastectomies during admission".

Intuitively, to better solve this domain-specific retrieval problem, we need to understand the requirements specified in a query and identify the documents satisfying these requirements based on their semantic meanings. In the past decades, significant efforts have been put on constructing biomedical knowledge bases (Aronson and Lang, 2010; Lipscomb, 2000; Corporation, 1999) and developing natural language processing (NLP) tools, such as MetaMap, to utilize the information from the knowledge bases (Aronson, 2001; McInnes et al., 2009). These efforts make it possible to map free text to concepts and use these concepts to represent queries and documents.

Indeed, *concept-based representation* is one of the commonly used approaches that leverage knowledge bases to improve the retrieval performance (Limsopatham et al., 2013d; Limsopatham et al., 2013b). The basic idea is to represent both queries and documents as "bags of concepts", where the concepts are identified based on the information from the knowledge bases. This method has been shown to be more effective than traditional term-based representation in the medical record retrieval because of its ability to handle the ambiguity in the medical terminology. However, this method also suffers the limitation that its effectiveness depends on the accuracy of the concept mapping results. As a result, directly applying existing weighting strategies might lead to non-optimal retrieval performance.

In this paper, to address the limitation caused by the inaccurate concept mapping results, we propose to regularize the weighting strategies in the concept-based representation methods. Specifically, by applying the axiomatic approaches (Fang and Zhai, 2005), we analyze the retrieval func-

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 603–612,
Baltimore, Maryland, USA, June 23-25 2014.

tions with concept-based representation and find that they may violate some reasonable retrieval constraints. We then propose two concept-based weighting regularization methods so that the regularized retrieval functions would satisfy the retrieval constraints and achieve better retrieval performance. Experimental results over two TREC collections show that both proposed concept-based weighting regularization methods can improve the retrieval performance, and their performance is comparable with the best systems of the TREC Medical Records tracks (Voorhees and Tong, 2011; Voorhees and Hersh, 2012).

Many NLP techniques have been developed to understand the semantic meaning of textual information, and are often applied to improve the search accuracy. However, due to the inherent ambiguity of natural languages, the results of NLP tools are not perfect. One of our contributions is to present a general methodology that can be used to adjust existing IR techniques based on the inaccurate NLP results.

2 Related Work

The Medical Records track of the Text REtrieval Conference (TREC) provides a common platform to study the medical records retrieval problem and evaluate the proposed methods (Voorhees and Tong, 2011; Voorhees and Hersh, 2012).

Concept-based representation has been studied for the medical record retrieval problem (Limsopatham et al., 2013d; Limsopatham et al., 2013b; Limsopatham et al., 2013a; Qi and Laquerre, 2012; Koopman et al., 2011; Koopman et al., 2012). For example, Qi and Laquerre used MetaMap to generate the concept-based representation and then apply a vector space retrieval model for ranking, and their results are one of the top ranked runs in the TREC 2012 Medical Records track (Qi and Laquerre, 2012). To further improve the performance, Limsopatham et al. proposed a task-specific representation, i.e., using only four types of concepts (symptom, diagnostic test, diagnosis and treatment) in the concept-based representation and a query expansion method based on the relationships among the medical concepts (Limsopatham et al., 2013d; Limsopatham et al., 2013a). Moreover, they also proposed a learning approach to combine both term-based and concept-based representation to further improve the performance (Limsopatham et al., 2013b).

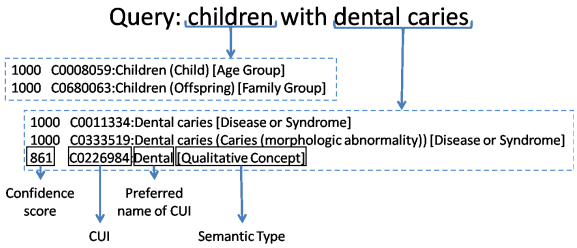


Figure 1: Example of MetaMap result for a query.

Our work is also related to domain-specific IR (Yan et al., 2011; Lin and Demner-Fushman, 2006; Zhou et al., 2007). For example, Yan et al. proposed a granularity-based document ranking model that utilizes ontologies to identify document concepts. However, none of the previous work has studied how to regularize the weight of concepts based on their relations.

It is well known that the effectiveness of a retrieval function is closely related to the weighting strategies (Fang and Zhai, 2005; Singhal et al., 1996). Various term weighting strategies have been proposed and studied for the term-based representation (Amati and Van Rijsbergen, 2002; Singhal et al., 1996; Robertson et al., 1996). However, existing studies on concept-based representation still used weighting strategies developed for term-based representation such as vector space models (Qi and Laquerre, 2012) and divergence from randomness (DFR) (Limsopatham et al., 2013a) and did not take the inaccurate concept mapping results into consideration. Compared with previous work, we focus on addressing the limitation caused by the inaccurate concept mapping. Note that our efforts are orthogonal to existing work, and it is expected to bring additional improvement to the retrieval performance.

3 Concept-based Representation for Medical Records Retrieval

3.1 Problem Formulation

We follow the problem setup used in the TREC medical record track (Voorhees and Tong, 2011; Voorhees and Hersh, 2012). The task is to retrieve relevant patient visits with respect to a query. Since each visit can be associated with multiple medical records, the relevance of a visit is related to the relevance of individual associated medical records. Existing studies computed the relevance

scores at either visit-level, where all the medical records of a visit are merged into a visit document (Demner-Fushman et al., 2012; Limsopatham et al., 2013c), or record-level, where we can first compute the relevance score of individual records and then aggregate their scores as the relevance score of a visit (Limsopatham et al., 2013c; Zhu and Carterette, 2012; Limsopatham et al., 2013d). In this paper, we focus on the visit-level relevance because of its simplicity. In particular, given a patient's visit, all the medical records generated from this visit are merged as a document. Note that our proposed concept-weighting strategies can also be easily applied to record-level relevance modeling.

Since the goal is to retrieve medical records of patients that satisfying requirements specified in a query, the relevance of medical records should be modeled based on how well they match all the requirements (i.e., aspects) specified in the queries.

3.2 Background: UMLS and MetaMap

Unified Medical Language System (UMLS) is a metathesaurus containing information from more than 100 controlled medical terminologies such as the Systematized Nomenclature of Medicine Clinical Terms (SNOMED-CT) and Medical Subject Headings (MeSH). Specifically, it contains the information about over 2.8 million biomedical concepts. Each concept is labeled with a Concept Unique Identifier (CUI) and has a preferred name and a semantic type.

Moreover, NLP tools for utilizing the information from UMLS have been developed. In particular, MetaMap (Aronson, 2001) can take a text string as the input, segment it into phrases, and then map each phrase to *multiple* UMLS CUIs with confidence scores. The confidence score is an indicator of the quality of the phrase-to-concept mapping by MetaMap. It is computed by four metrics: centrality, variation, coverage and cohesiveness (Aronson, 2001). These four measures try to evaluate the mapping from different angles, such as the involvement of the central part, the distance of the concept to the original phrase, and how well the concept matches the phrase. The maximum confidence in MetaMap is 1000.

Figure 1 shows the MetaMap results for an example query "children with dental caries". Two *query aspects*, i.e., "children" and "dental caries", are identified. Each of them is mapped to multiple concepts, and each concept is associated with the confidence score as well as more detailed information about this concept.

3.3 Concept-based Representation

Traditional retrieval models are based on "bag of terms" representation. One limitation of this representation is that relevance scores are computed based on the matching of terms rather than the meanings. As a result, the system may fail to retrieve the relevant documents that do not contain any query terms.

To overcome this limitation, concept-based representation has been proposed to bridge the vocabulary gap between documents and queries (Qi and Laquerre, 2012; Limsopatham et al., 2013b; Koopman et al., 2012). In particular, MetaMap is used to map terms from queries and documents (e.g., medical records) to the semantic concepts from biomedical knowledge bases such as UMLS. Within the concept-based representation, the query can then be represented as a bag of all the generated CUIs in the MetaMap results. For example, the query from Figure 1 can be represented as $\{C0008059, C0680063, C0011334, C0333519, C0226984\}$. Documents can be represented in a similar way.

After converting both queries and documents to concept-based representations using MetaMap, previous work applied existing retrieval functions such as vector space models (Singhal et al., 1996) to rank the documents. Note that when referring to existing retrieval functions in the paper, they include traditional keyword matching based functions such as pivoted normalization (Singhal et al., 1996), Okapi (Robertson et al., 1996), Dirichlet prior (Zhai and Lafferty, 2001) and basic axiomatic functions (Fang and Zhai, 2005).

4 Weighting Strategies for Concept-based Representation

4.1 Motivation

Although existing retrieval functions can be directly applied to concept-based representation, they may lead to non-optimal performance. This is mainly caused by the fact that MetaMap may generate more than one mapped concepts for an aspect, i.e., a semantic unit in the text.

Ideally, an aspect will be mapped to only *one* concept, and different concepts would represent different semantic meanings. Under such a situ-

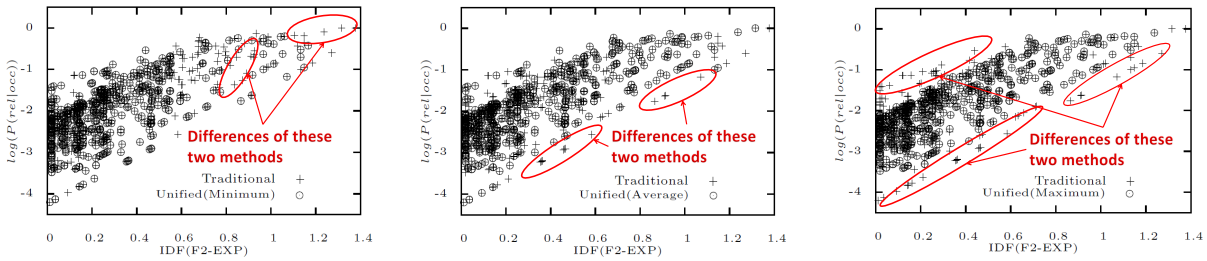


Figure 2: Exploratory data analysis (From left to right are choosing minimum, average and maximum IDF concepts as the representing concepts, respectively. The removed concepts are highlighted in the figures.).

ation, traditional retrieval functions would likely work well and generate satisfying retrieval performance since the relations among concepts are independent which is consistent with the assumptions made in traditional IR (Manning et al., 2008).

However, the mapping results generated by MetaMap are not perfect. Although MetaMap is able to rank all the candidate concepts with the confidence score and pick the most likely one, the accuracy is not very high. In particular, our preliminary results show that turning on the disambiguation functionality provided by MetaMap (i.e., returning only the most likely concept for each query) could lead to worse retrieval performance than using all the candidate mappings. Thus, we use the one-to-many mapping results generated by MetaMap, in which each aspect can be mapped to multiple concepts.

Unfortunately, such one-to-many concept mappings could hinder the retrieval performance in the following two ways.

- The multiple concepts generated from the same aspect are related, which is inconsistent with the independence assumption made in the existing retrieval functions (Manning et al., 2008). For example, as shown in Figure 1, "dental caries" is mapped to three concepts. It is clear that the concepts are related, but existing retrieval functions are unable to capture their relations and would compute the weight of each concept independently.

- The one-to-many mapping results generated by MetaMap could arbitrarily inflate the weights of some query aspects. For example, as shown in Figure 1, query aspect "children" is mapped to 2 concepts while "dental caries" is mapped to 3 concepts. In the existing retrieval functions, term occurrences are important relevance signals. However, when converting the text to concepts representation using MetaMap, the occurrences of the concepts are determined by not only the original term occurrences, a good indicator of relevance, but also the number of mapped concepts, which is determined by MetaMap and has nothing to do with the relevance status. As a result, the occurrences of concepts might not be a very accurate indicator of importance of the corresponding query aspect.

To address the limitations caused by the inaccurate mapping results, we propose to apply axiomatic approaches (Fang and Zhai, 2005) to regularize the weighting strategies for concept-based representation methods. In particular, we first formalize retrieval constraints that any reasonable concept-based representation methods should satisfy and then discuss how to regularize the existing weighting strategies to satisfy the constraints and improve the retrieval performance.

We first explain the notations used in this section. Q and D denote a query and a document with the concept-based representation. $S(Q, D)$ is the relevance score of D with respect to Q. e_i denotes a concept, and $\mathcal{A}(e)$ denotes the query aspect associated with e, i.e., a set of concepts that are mapped to the same phrases as e by using MetaMap. $i(e)$ is the normalized confidence score of the mapping for concept e generated by MetaMap. $c(e, D)$ denotes the occurrences of concept e in document D, $df(e)$ denotes the number of documents containing e. $|D|$ is the document length of D. $Imp_c(e)$ is the importance of the concept such as the concept IDF value, and $Imp_A(\mathcal{A})$ is the importance of the aspect.

4.2 Unified concept weighting regularization

We now discuss how to address the first challenge, i.e,. how to regularize the weighting strategy so that we can take into consideration the fact that concepts associated with the same query aspect are not independent. We call a concept is a *variant* of another one if both of them are associated with the same aspect.

Intuitively, given a query with two aspects, a document covering both aspects should be ranked higher than those covering only one aspect. We can formalize the intuition in the concept-based representation as the following constraint.

Unified Constraint: Let query be $Q = \{e_1, e_2, e_3\}$, and we know that e_2 is a variant of e_3. Assume we have two documents D_1 and D_2 with the same document length, i.e., $|D_1| = |D_2|$. If we know that $c(e_1, D_1) = c(e_3, D_2) > 0$, $c(e_1, D_2) = c(e_3, D_1) = 0$ and $c(e_2, D_1) = c(e_2, D_2) > 0$, then $S(Q, D_1) > S(Q, D_2)$.

It is clear that existing retrieval functions would violate this constraint since they ignore the relations among concepts.

One simple strategy to fix this problem is to merge all the concept variants as a single concept and select one representative concept to replace all occurrences of other variants in both queries and documents. By merging the concepts together, we are aiming to purify the concepts and make the similar concepts centralized so that the assumption that all the concepts are independent would hold.

Formally, the adjusted occurrences of a concept e in a document D is shown as follows:

$$c_{mod}(e, D) = \begin{cases} \sum_{e' \in EC(e)} c(e', D) & e = Rep(EC(e)) \\ 0 & e \neq Rep(EC(e)) \end{cases} \quad (1)$$

where $c(e, D)$ is the original occurrence of concept e in document D, $EC(e)$ denotes a set of all the variants of e including itself (i.e., all the concepts with the same preferred name as e), and $Rep(EC(e))$ denotes the representative concept from $EC(e)$.

It is trivial to prove that, with such changes, existing retrieval functions would satisfy the above constraint since the constraint implies TFC2 constraint defined in the previous study (Fang et al., 2004).

Now the remaining question is how to select the representative concept from all the variants. There are three options: select the concept with the maximum IDF, average IDF, or minimum IDF. We conduct exploratory data analysis on these three options. In particular, for each option, we generate a plot indicating the correlation between the IDF value of a concept and the relevance probability of the concept (i.e., the probability that a document containing the concept is relevant). Note that both original and replaced IDF values are shown in the plot for each option. Figure 2 shows the results. It is clear that the right plot (i.e., selecting the concept with the maximum IDF as the representative concept) is the best choice since the changes make the points less scattered. In fact, this can also be confirmed by experimental results as reported in Table 5. Thus, we use the concept with the maximum IDF value as the representative concept of all the variants.

4.3 Balanced concept weighting regularization

We now discuss how to address the second challenge, i.e., how to regularize the weighting strategy to deal with the arbitrarily inflated statistics caused by the one-to-many mappings.

The arbitrary inflation could impact the importance of the query aspects. For example, as shown in Figure 1, one aspect is mapped to two concepts while the other is mapped to three. Moreover, it could also impact the accuracy of the concept IDF values. Consider "colonoscopies" and "adult", it is clear that the first term is more important than the second one, which is consistent with their term IDF values, i.e., 7.52 and 2.92, respectively. However, with the concept-based representation, the IDF value of the concept "colonoscopies"(C0009378) is 2.72, which is even smaller than that of concept "adult" (C1706450), i.e., 2.92.

To fix the negative impact on query aspects, we could leverage the findings in the previous study (Zheng and Fang, 2010) and regularize the weighting strategy based on the length of query aspects to favor documents covering more query aspects. Since each concept mapping is associated with a confidence score, we can incorporate them into the regularization function as follows:

$$f(e, Q) = (1 - \alpha) + \alpha \cdot \Big(\frac{\sum_{e' \in Q} i(e')}{\sum_{e'' \in \mathcal{A}(e)} i(e'')} \Big), \quad (2)$$

where $i(e)$ is the normalized confidence score of concept e generated by MetaMap, and α is a parameter between 0 and 1 to control the effect of the regularization. When α is set to 0, there is no regularization. This regularization function aims to

penalize the weight of concept e based on its variants as well as the concepts from other aspects. In particular, a concept would receive more penalty (i.e., its weight will be decreased more) when it has more variants and the mappings of these variants are more accurate.

To fix the negative impact on the concept IDF values, we propose to regularize the weighting based on the importance of the query aspect. This regularization can be formalized as the following constraint.

Balanced Constraint: Let Q be a query with two concepts and the concepts are associated with different aspects, i.e., $Q = \{e_1, e_2\}$, and $\mathcal{A}(e_1) \neq \mathcal{A}(e_2)$. Assume D_1 and D_2 are two documents with the same length, i.e., $|D_1| = |D_2|$, and they cover different concepts with the same occurrences, i.e., $c(e_1, D_1) = c(e_2, D_2) > 0$ and $c(e_2, D_1) = c(e_1, D_2) = 0$. If we know $Imp_c(e_1) = Imp_c(e_2)$ and $Imp_A(\mathcal{A}(e_1)) < Imp_A(\mathcal{A}(e_2))$, then we have $S(Q, D_1) < S(Q, D_2)$.

This constraint requires that the relevance score of a document should be affected by not only the importance of the concepts but also the importance of the associated query aspect. In a way, the constraint aims to counteract the arbitrary statistics inflation caused by MetaMap results and balance the weight among concepts based on the importance of the associated query aspects. And it is not difficult to show that existing retrieval functions violate this constraint.

Now the question is how to revise the retrieval functions to make them satisfy this constraint. We propose to incorporate the importance of query aspect into the previous regularization function in Equation (2) as follows:

$$f(e, Q) = (1-\alpha) + \alpha \cdot \left(\frac{\sum_{e' \in Q} i(e')}{\sum_{e'' \in \mathcal{A}(e)} i(e'')} \right) \cdot Imp_A(\mathcal{A}(e)). \quad (3)$$

Note that $Imp_A(\mathcal{A}(e))$ is the importance of a query aspect and can be estimated based on the terms from the query aspect. In this paper, we use the maximum term IDF value from the aspect to estimate the importance, which performs better than using minimum and average IDF values as shown in the experiments (i.e., Table 6). We plan to study other options in the future work.

4.4 Discussions

Both proposed regularization methods can be combined with any existing retrieval functions. In this paper, we focus on one of the state of the art weighting strategies, i.e., F2-EXP function derived from axiomatic retrieval model (Fang and Zhai, 2005), and explain how to incorporate the regularization methods into the function.

The original F2-EXP retrieval function is shown as follows:

$$S(Q, D) = \sum_{e \in Q \cap D} c(e, Q) \cdot (\frac{N}{df(e)})^{0.35} \cdot \frac{c(e, D)}{c(e, D) + b + \frac{b \times |D|}{avdl}} \quad (4)$$

where b is a parameter control the weight of the document length normalization.

With the unified concept weighting regularization, the revised function based on F2-EXP function, i.e., *Unified*, is shown as follows:

$$S(Q, D) = \sum_{e \in Q \cap D} c_{mod}(e, Q) \cdot (\frac{N}{df(t)})^{0.35} \cdot \frac{c_{mod}(e, D)}{c_{mod}(e, D) + b + \frac{b \times |D|}{avdl}} \quad (5)$$

where $c_{mod}(e, D)$ and $c_{mod}(e, Q)$ denote the modified occurrences as shown in Equation (1). It can be shown that this function satisfies the unified constraint but violates the balanced constraint.

Following the similar strategy used in the previous study (Zheng and Fang, 2010), we can further incorporate the regularization function proposed in Equation (3) to the above function to make it satisfy the balanced constraint as follows:

$$S(Q, D) = \sum_{e \in Q \cap D} c_{mod}(e, Q) \cdot (\frac{N}{df(t)})^{0.35} \cdot f(e, Q) \cdot \frac{c_{mod}(e, D)}{c_{mod}(e, D) + b + \frac{b \times |D|}{avdl}} \quad (6)$$

where $f(e, Q)$ is the newly proposed regularization function as shown in Equation (3). This method is denoted as *Balanced*, and can be shown that it satisfies both constraints.

Table 1: Statistics of collections.

	# of unique tokens	AvgDL	AvgQL11	AvgQL12
Term	263,356	2,659	10.23	8.82
Concept	58,192	2,673	8.79	7.81

5 Experiments

5.1 Experiment Setup

We conduct experiments using two data sets from the TREC Medical Records track 2011 and 2012.

Table 2: Description of Methods

Name	Representation	Ranking strategies
Term-BL	Term	F2-EXP (i.e., Equation (4))
Concept-BL	Concept (i.e., Section 3.3)	F2-EXP (i.e., Equation (4))
TSConcept-BL	Task specific concept ((Limsopatham et al., 2013d))	F2-EXP (i.e., Equation (4))
Unified	Concept (i.e., Section 4.2)	F2-EXP + Unified (i.e., Equation (5))
Balanced	Concept (i.e., Section 4.3)	F2-EXP + Balanced (i.e., Equation (6))

Table 3: Performance under optimized parameter settings

	Med11		**Med12**	
	MAP	bpref	infNDCG	infAP
Term-BL	0.3474	0.4727	0.4695	0.2106
Concept-BL	0.3967	0.5476	0.5243	0.2497
TSConcept-BL	0.3964	0.5329	0.5283	0.2694
Unified	0.4235^{T}	0.5443^{T}	0.5416^{T}	0.2586^{T}
Balanced	$\mathbf{0.4561}^{T,C,TS}$	$\mathbf{0.5697}^{T,C,TS}$	$\mathbf{0.5767}^{T,C,TS}$	$\mathbf{0.2859}^{T,C,TS}$

The data sets are denoted as *Med11* and *Med12*. Both data sets used the same document collection with 100,866 medical records, each of which is associated with a unique patient visit to the hospital or emergency department. Since the task is to retrieve relevant visits, we merged all the records from a visit to form a single document for the visit, which leads to 17,198 documents in the collection. There are 34 queries in *Med11* and 47 in *Med12*. These queries were developed by domain experts based on the "inclusion criteria" of a clinical study (Voorhees and Tong, 2011; Voorhees and Hersh, 2012).

After applying MetaMap to both documents and queries, we can construct a concept-based collection. Since documents are often much longer, we can first segment them into sentences, get the mapping results for each sentence, and then merge them together to generate the concept-based representation for the documents.

Table 1 compares the statistics of the term-based and the concept-based collections, including the number of unique tokens in the collection (i.e., the number of terms for term-based representation and the number of concepts for concept-based representation), the average number of tokens in the documents (AvgDL) and the average number of tokens in the queries for these two collections (AvgQL11 and AvgQL12). It is interesting to see that the number of unique tokens is much smaller when using the concept-based indexing. This is expected since terms are semantically related and a group of related terms would be mapped to one semantic concept. Moreover, we observe that the document length and query length are similar for both collections. This is caused by the fact that concepts are related and the MetaMap would map an aspect to multiple related concepts.

Table 2 summarizes the methods that we compare in the experiments. Following the evaluation methodology used in the medical record track, we use MAP@1000 as the primary measure for *Med11* and also report bpref. For *Med12*, we take infNDCG@100 as the primary measure and also report infAP@100. Different measures were chosen for these two sets mainly because different pooling strategies were used to create the judgment pools (Voorhees and Hersh, 2012).

5.2 Performance Comparison

Table 3 shows the performance under optimized parameter settings for all the methods over both data sets. The performance is optimized in terms of MAP in *Med11*, and infNDCG in *Med12*, respectively. α and b are tuned from 0 to 1 with the step 0.1. Note that ^{T}, ^{C} and ^{TS} indicate improvement over Term-BL, Concept-BL and TSConcept-BL is statistically significant at 0.05 level based on Wilcoxon signed-rank test, respectively.

Results show that *Balanced* method can significantly improve the retrieval performance over both collections. *Unified* method outperforms the baseline methods in terms of the primary measure on both collections, although it fails to improve the infAP on *Med12* for one baseline method. It is not surprising to see that *Balanced* method is more effective than *Unified* since the former satisfies both of the proposed retrieval constraints while the lat-

Table 4: Testing Performance

Trained on	**Med12**		**Med11**	
Tested on	**Med11**		**Med12**	
Measures	MAP	bpref	infNDCG	infAP
Term-BL	0.3451	0.4682	0.4640	0.2040
Concept-BL	0.3895	0.5394	0.5194	0.2441
TSConcept-BL	0.3901	0.5286	0.5208	0.2662
Unified	$0.4176^{T,C}$	0.5391^{T}	0.5346^{T}	0.2514^{T}
Balanced	$\mathbf{0.4497}^{T,C,TS}$	$\mathbf{0.5627}^{T,C,TS}$	$\mathbf{0.5736}^{T,C,TS}$	$\mathbf{0.2811}^{T,C,TS}$

ter satisfies only one. Finally, we noticed that the performance difference between TSConcept-BL and Concept-BL is not as significant as the ones reported in the previous study (Limsopatham et al., 2013d), which is probably caused by the difference of problem set up (i.e., record-level vs. visit-level as discussed in Section 3.1).

We also conduct experiments to train parameters on one collection and compare the testing performance on the other collection. The results are summarized in Table 4. Clearly, *Balanced* is still the most effective regularization method. The testing performance is very close to the optimal performance, which indicates that the proposed methods are robust with respect to the parameter setting.

Moreover, we would like to point out that the testing performance of *Balanced* is comparable to the top ranked runs from the TREC Medical records track. For example, the performance of the best automatic system in *Med11* (e.g., CengageM11R3) is 0.552 in terms of bpref, while the performance of the best automatic system in *Med12* (e.g., udelSUM) is 0.578 in terms of infNDCG. Note that the top system of *Med12* used multiple external resources such as Wikipedia and Web, while we did not use such resources. Moreover, our performance might be further improved if we apply the result filtering methods used by many TREC participants (Leveling et al., 2012).

Table 5: Selecting representative concepts

	MAP	**bpref**
Unified (i.e., Unified-max)	**0.4235**	**0.5443**
Unified-min	0.3894	0.5202
Unified-avg	0.4164	0.5303

5.3 More Analysis

In the *Unified* method, we chose the concept with the maximum IDF as the representative concept among all the variants. We now conduct experiments on *Med11* to compare its performance with those of using average IDF and minimum IDF ones as the representative concept. The results are shown in Table 5. It is clear that using maximum IDF is the best choice, which is consistent with our observation from the data exploratory analysis shown in Figure 2.

Table 6: Estimating query aspect importance

	MAP	**bpref**
Balanced (i.e., Balanced-max)	**0.4561**	**0.5697**
Balanced-min	0.4216	0.5484
Balanced-avg	0.4397	0.5581

Table 7: Regularization components in *Balanced*

	MAP	**bpref**
Balanced	**0.4561**	**0.5697**
Confidence only	0.4294	0.5507
Importance only	0.4373	0.5598

In the *Balanced* method, we used the maximum IDF value to estimate the query importance. We also conduct experiments to compare its performance with those using the minimum and average IDF values. Table 6 summarizes the results, and shows that using the maximum IDF value performs better than the other choices.

As shown in Equation (3), the *Balanced* method regularizes the weights through two components: (1) normalized confidence score of each aspect, i.e., $\frac{\sum_{e' \in Q} i(e')}{\sum_{e'' \in \mathcal{A}(e)} i(e'')}$; and (2) the importance of the query aspect, i.e., $Imp_A(\mathcal{A}(e))$. To examine the effectiveness of each component, we conduct experiments using the modified *Balanced* method with only one of the components. The results are shown in Table 7. It is clear that both components are essential to improve the retrieval performance.

Finally, we report the performance improvement of the proposed methods over the *Concept-BL* for each query in Figure 3. Clearly, both of the

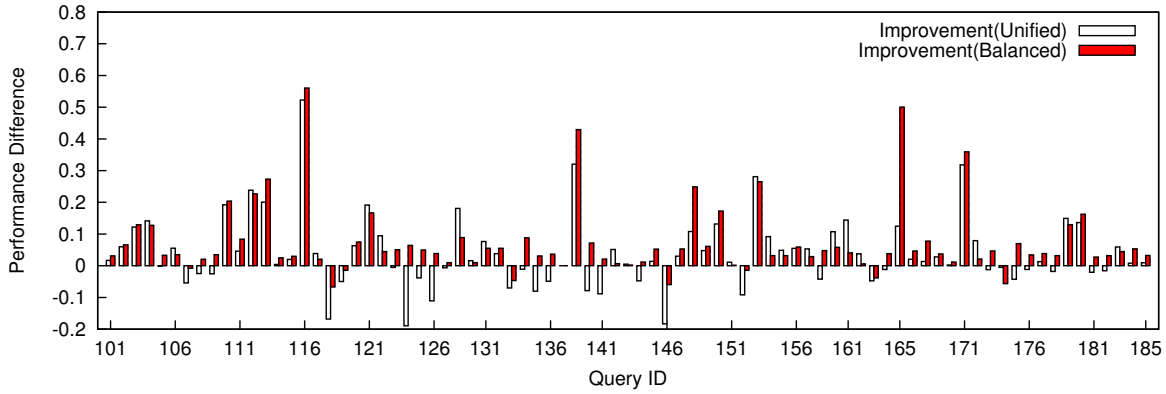


Figure 3: Improvement of proposed methods (Compared with the Concept-BL method).

proposed methods can improve the effectiveness of most queries, and the *Balanced* method is more robust than the *Unified* method.

6 Conclusions and Future Work

Medical record retrieval is an important domain-specific IR problem. Concept-based representation is an effective approach to dealing with ambiguity terminology in medical domain. However, the results of the NLP tools used to generate the concept-based representation are often not perfect. In this paper, we present a general methodology that can use axiomatic approaches as guidance to regularize the concept weighting strategies to address the limitations caused by the inaccurate concept mapping and improve the retrieval performance. In particular, we proposed two weighting regularization methods based on the relations among concepts. Experimental results show that the proposed methods can significantly outperform existing retrieval functions.

There are many interesting directions for our future work. First, we plan to study how to automatically predict whether to use concept-based indexing based on the quality of MetaMap results, and explore whether the proposed methods are applicable for other entity linking methods. Second, we will study how to leverage other information from knowledge bases to further improve the performance. Third, more experiments could be conducted to examine the effectiveness of the proposed methods when using other ranking strategies. Finally, it would be interesting to study how to follow the proposed methodology to study other domain-specific IR problems.

References

Gianni Amati and Cornelis Joost Van Rijsbergen. 2002. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM TOIS*.

Alan R. Aronson and François-Michel Lang. 2010. An overview of metamap: historical perspective and recent advances. *JAMIA*, 17(3):229–236.

Alan R. Aronson. 2001. Effective mapping of biomedical text to the UMLS Metathesaurus: the MetaMap program. In *Proceedings of AMIA Symposium*.

Practice Management Information Corporation. 1999. *ICD-9-CM: International Classification of Diseases, 9th Revision, Clinical Modification, 5th Edition*. Practice Management Information Corporation.

Dina Demner-Fushman, Swapna Abhyankar, Antonio Jimeno-Yepes, Russell Loane, Francois Lang, James G. Mork, Nicholas Ide, and Alan R. Aronson. 2012. NLM at TREC 2012 Medical Records Track. In *Proceedings of TREC 2012*.

Hui Fang and ChengXiang Zhai. 2005. An exploration of axiomatic approaches to information retrieval. In *Proceedings of SIGIR'05*.

Hui Fang, Tao Tao, and ChengXiang Zhai. 2004. A formal study of information retrieval heuristics. In *Proceedings of SIGIR'04*.

Charles P. Friedman, Adam K. Wong, and David Blumenthal. 2010. Achieving a nationwide learning health system. *Science Translational Medicine*.

Beval Koopman, Michael Lawley, and Peter Bruza. 2011. AEHRC & QUT at TREC 2011 Medical Track : a concept-based information retrieval approach. In *Proceedings of TREC'11*.

Bevan Koopman, Guido Zuccon, Anthony Nguyen, Deanne Vickers, Luke Butt, and Peter D. Bruza.

2012. Exploiting SNOMED CT Concepts & Relationships for Clinical Information Retrieval: Australian e-Health Research Centre and Queensland University of Technology at the TREC 2012 Medical Track. In *Proceedings of TREC'12*.

Johannes Leveling, Lorraine Goeuriot, Liadh Kelly, and Gareth J. F. Jones. 2012. DCU@TRECMed 2012: Using adhoc Baselines for Domain-Specific Retrieval. In *Proceedings of TREC 2012*.

Nut Limsopatham, Craig Macdonald, and Iadh Ounis. 2013a. Inferring conceptual relationships to improve medical records search. In *Proceedings of OAIR'13*.

Nut Limsopatham, Craig Macdonald, and Iadh Ounis. 2013b. Learning to combine representations for medical records search. In *Proceedings of SIGIR'13*.

Nut Limsopatham, Craig Macdonald, and Iadh Ounis. 2013c. Learning to selectively rank patients' medical history. In *Proceedings of CIKM'13*.

Nut Limsopatham, Craig Macdonald, and Iadh Ounis. 2013d. A task-specific query and document representation for medical records search. In *Proceedings of ECIR'13*.

Jimmy Lin and Dina Demner-Fushman. 2006. The role of knowledge in conceptual retrieval: a study in the domain of clinical medicine. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 99–106, New York, NY, USA. ACM.

Carolyn E Lipscomb. 2000. Medical Subject Headings (MeSH). *The Medical Library Association*.

Christopher D. Manning, P. Raghavan, and H. Schutze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Bridget T. McInnes, Ted Pedersen, and Serguei V. S. Pakhomov. 2009. UMLS-Interface and UMLS-Similarity : Open Source Software for Measuring Paths and Semantic Similarity. In *Proceedings of AMIA Symposium*.

Yanjun Qi and Pierre-Francois Laquerre. 2012. Retrieving Medical Records: NEC Labs America at TREC 2012 Medical Record Track. In *Proceedings of TREC 2012*.

S.E. Robertson, S. Walker, S. Jones, M.M. Hancock-Beaulieu, and M. Gatford. 1996. Okapi at TREC-3. pages 109–126.

Charles Safran, Meryl Bloomrosen, W. Edward Hammond, Steven Labkoff, Suzanne Markel-Fox, Paul C. Tang, and Don E. Detmer. 2007. White paper: Toward a national framework for the secondary use of health data: An american medical informatics association white paper. *JAMIA*, 14(1):1–9.

Amit Singhal, Chris Buckley, and Mandar Mitra. 1996. Pivoted document length normalization. In *Proceedings of SIGIR'96*.

Ellen M. Voorhees and William Hersh. 2012. Overview of the TREC 2012 Medical Records Track. In *Proceedings of TREC 2012*.

Ellen M. Voorhees and Richard M. Tong. 2011. Overview of the TREC 2011 Medical Records Track. In *Proceedings of TREC 2011*.

Xin Yan, Raymond Y.K. Lau, Dawei Song, Xue Li, and Jian Ma. 2011. Toward a semantic granularity model for domain-specific information retrieval. *ACM TOIS*.

Chengxiang Zhai and John Lafferty. 2001. A study of smoothing methods for language models applied to Ad Hoc information retrieval. In *Proceedings of SIGIR'01*.

Wei Zheng and Hui Fang. 2010. Query aspect based term weighting regularization in information retrieval. In *Proceedings of ECIR'10*.

Wei Zhou, Clement Yu, Neil Smalheiser, Vetle Torvik, and Jie Hong. 2007. Knowledge-intensive conceptual retrieval and passage extraction of biomedical literature. In *Proceedings of SIGIR'07*.

Dongqing Zhu and Ben Carterette. 2012. Combining multi-level evidence for medical record retrieval. In *Proceedings of SHB'12*.

Learning to Predict Distributions of Words Across Domains

Danushka Bollegala
Department of Computer Science
University of Liverpool
Liverpool,
L69 3BX, UK
danushka.bollegala@liverpool.ac.uk

David Weir
Department of Informatics
University of Sussex
Falmer, Brighton,
BN1 9QJ, UK
d.j.weir@sussex.ac.uk

John Carroll
Department of Informatics
University of Sussex
Falmer, Brighton,
BN1 9QJ, UK
j.a.carroll@sussex.ac.uk

Abstract

Although the distributional hypothesis has been applied successfully in many natural language processing tasks, systems using distributional information have been limited to a single domain because the distribution of a word can vary between domains as the word's predominant meaning changes. However, if it were possible to *predict* how the distribution of a word changes from one domain to another, the predictions could be used to adapt a system trained in one domain to work in another. We propose an unsupervised method to predict the distribution of a word in one domain, given its distribution in another domain. We evaluate our method on two tasks: cross-domain part-of-speech tagging and cross-domain sentiment classification. In both tasks, our method significantly outperforms competitive baselines and returns results that are statistically comparable to current state-of-the-art methods, while requiring no task-specific customisations.

1 Introduction

The Distributional Hypothesis, summarised by the memorable line of Firth (1957) – *You shall know a word by the company it keeps* – has inspired a diverse range of research in natural language processing. In such work, a word is represented by the distribution of other words that co-occur with it. Distributional representations of words have been successfully used in many language processing tasks such as entity set expansion (Pantel et al., 2009), part-of-speech (POS) tagging and chunking (Huang and Yates, 2009), ontology learning (Curran, 2005), computing semantic textual similarity (Besançon et al., 1999), and lexical inference (Kotlerman et al., 2012).

However, the distribution of a word often varies from one domain[1] to another. For example, in the domain of portable computer reviews the word *lightweight* is often associated with positive sentiment bearing words such as *sleek* or *compact*, whereas in the movie review domain the same word is often associated with negative sentiment-bearing words such as *superficial* or *formulaic*. Consequently, the distributional representations of the word *lightweight* will differ considerably between the two domains. In this paper, given the distribution $\boldsymbol{w}_{\mathcal{S}}$ of a word w in the source domain $\mathcal{S}$, we propose an unsupervised method for *predicting* its distribution $\boldsymbol{w}_{\mathcal{T}}$ in a different target domain $\mathcal{T}$.

The ability to predict how the distribution of a word varies from one domain to another is vital for numerous adaptation tasks. For example, unsupervised cross-domain sentiment classification (Blitzer et al., 2007; Aue and Gamon, 2005) involves using sentiment-labeled user reviews from the source domain, and unlabeled reviews from both the source and the target domains to learn a sentiment classifier for the target domain. Domain adaptation (DA) of sentiment classification becomes extremely challenging when the distributions of words in the source and the target domains are very different, because the features learnt from the source domain labeled reviews might not appear in the target domain reviews that must be classified. By predicting the distribution of a word across different domains, we can find source domain features that are similar to the features in target domain reviews, thereby reducing the mismatch of features between the two domains.

We propose a two-step unsupervised approach to predict the distribution of a word across domains. First, we create two lower dimensional la-

[1] In this paper, we use the term *domain* to refer to a collection of documents about a particular topic, for example reviews of a particular kind of product.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 613–623,
Baltimore, Maryland, USA, June 23-25 2014.

tent feature spaces separately for the source and the target domains using Singular Value Decomposition (SVD). Second, we learn a mapping from the source domain latent feature space to the target domain latent feature space using Partial Least Square Regression (PLSR). The SVD smoothing in the first step both reduces the data sparseness in distributional representations of individual words, as well as the dimensionality of the feature space, thereby enabling us to efficiently and accurately learn a prediction model using PLSR in the second step. Our proposed cross-domain word distribution prediction method is unsupervised in the sense that it does not require any labeled data in either of the two steps.

Using two popular multi-domain datasets, we evaluate the proposed method in two prediction tasks: (a) predicting the POS of a word in a target domain, and (b) predicting the sentiment of a review in a target domain. Without requiring any task specific customisations, systems based on our distribution prediction method significantly outperform competitive baselines in both tasks. Because our proposed distribution prediction method is unsupervised and task independent, it is potentially useful for a wide range of DA tasks such entity extraction (Guo et al., 2009) or dependency parsing (McClosky et al., 2010). Our contributions are summarised as follows:

- Given the distribution $\boldsymbol{w}_{\mathcal{S}}$ of a word w in a source domain $\mathcal{S}$, we propose a method for learning its distribution $\boldsymbol{w}_{\mathcal{T}}$ in a target domain $\mathcal{T}$.
- Using the learnt distribution prediction model, we propose a method to learn a cross-domain POS tagger.
- Using the learnt distribution prediction model, we propose a method to learn a cross-domain sentiment classifier.

To our knowledge, ours is the first successful attempt to learn a model that predicts the distribution of a word across different domains.

2 Related Work

Learning semantic representations for words using documents from a single domain has received much attention lately (Vincent et al., 2010; Socher et al., 2013; Baroni and Lenci, 2010). As we have already discussed, the semantics of a word varies across different domains, and such variations are not captured by models that only learn a single semantic representation for a word using documents from a single domain.

The POS of a word is influenced both by its context (*contextual bias*), and the domain of the document in which it appears (*lexical bias*). For example, the word *signal* is predominately used as a noun in MEDLINE, whereas it appears predominantly as an adjective in the Wall Street Journal (WSJ) (Blitzer et al., 2006). Consequently, a tagger trained on WSJ would incorrectly tag *signal* in MEDLINE. Blitzer et al. (2006) append the source domain labeled data with predicted pivots (i.e. words that appear in both the source and target domains) to adapt a POS tagger to a target domain. Choi and Palmer (2012) propose a cross-domain POS tagging method by training two separate models: a generalised model and a domain-specific model. At tagging time, a sentence is tagged by the model that is most similar to that sentence. Huang and Yates (2009) train a Conditional Random Field (CRF) tagger with features retrieved from a smoothing model trained using both source and target domain unlabeled data. Adding latent states to the smoothing model further improves the POS tagging accuracy (Huang and Yates, 2012). Schnabel and Schütze (2013) propose a training set filtering method where they eliminate shorter words from the training data based on the intuition that longer words are more likely to be examples of productive linguistic processes than shorter words.

The sentiment of a word can vary from one domain to another. In Structural Correspondence Learning (SCL) (Blitzer et al., 2006; Blitzer et al., 2007), a set of pivots are chosen using pointwise mutual information. Linear predictors are then learnt to predict the occurrence of those pivots, and SVD is used to construct a lower dimensional representation in which a binary classifier is trained. Spectral Feature Alignment (SFA) (Pan et al., 2010) also uses pivots to compute an alignment between domain specific and domain independent features. Spectral clustering is performed on a bipartite graph representing domain specific and domain independent features to find a lower-dimensional projection between the two sets of features. The cross-domain sentiment-sensitive thesaurus (SST) (Bollegala et al., 2011) groups together words that express similar sentiments in

different domains. The created thesaurus is used to expand feature vectors during train and test stages in a binary classifier. However, unlike our method, SCL, SFA, or SST do *not* learn a prediction model between word distributions across domains.

Prior knowledge of the sentiment of words, such as sentiment lexicons, has been incorporated into cross-domain sentiment classification. He et al. (2011) propose a joint sentiment-topic model that imposes a sentiment-prior depending on the occurrence of a word in a sentiment lexicon. Ponomareva and Thelwall (2012) represent source and target domain reviews as nodes in a graph and apply a label propagation algorithm to predict the sentiment labels for target domain reviews from the sentiment labels in source domain reviews. A sentiment lexicon is used to create features for a document. Although incorporation of prior sentiment knowledge is a promising technique to improve accuracy in cross-domain sentiment classification, it is complementary to our task of distribution prediction across domains.

The *unsupervised* DA setting that we consider does not assume the availability of labeled data for the target domain. However, if a small amount of labeled data is available for the target domain, it can be used to further improve the performance of DA tasks (Xiao et al., 2013; Daumé III, 2007).

3 Distribution Prediction

3.1 In-domain Feature Vector Construction

Before we tackle the problem of learning a model to predict the distribution of a word across domains, we must first compute the distribution of a word from a single domain. For this purpose, we represent a word w using unigrams and bigrams that co-occur with w in a sentence as follows.

Given a document H, such as a user-review of a product, we split H into sentences, and lemmatize each word in a sentence using the RASP system (Briscoe et al., 2006). Using a standard stop word list, we filter out frequent non-content unigrams and select the remainder as unigram features to represent a sentence. Next, we generate bigrams of word lemmas and remove any bigrams that consists only of stop words. Bigram features capture negations more accurately than unigrams, and have been found to be useful for sentiment classification tasks. Table 1 shows the unigram and bigram features we extract for a sentence using this procedure. Using data from a single domain, we construct a feature co-occurrence matrix $\mathbf{A}$ in which columns correspond to unigram features and rows correspond to either unigram or bigram features. The value of the element a_{ij} in the co-occurrence matrix $\mathbf{A}$ is set to the number of sentences in which the i-th and j-th features co-occur.

sentence	This is an interesting and well researched book
unigrams (surface)	this, is, an, interesting, and, well, researched, book
unigrams (lemma)	this, be, an, interest, and, well, research, book
unigrams (features)	interest, well, research, book
bigrams (lemma)	this+be, be+an, an+interest, interest+and, and+well, well+research, research+book
bigrams (features)	an+interest, interest+and, and+well, well+research, research+book

Table 1: Extracting unigram and bigram features.

Typically, the number of unique bigrams is much larger than that of unigrams. Moreover, co-occurrences of bigrams are rare compared to co-occurrences of unigrams, and co-occurrences involving a unigram and a bigram. Consequently, in matrix $\mathbf{A}$, we consider co-occurrences only between unigrams vs. unigrams, and bigrams vs. unigrams. We consider each row in $\mathbf{A}$ as representing the distribution of a feature (i.e. unigrams or bigrams) in a particular domain over the unigram features extracted from that domain (represented by the columns of $\mathbf{A}$). We apply Positive Pointwise Mutual Information (PPMI) to the co-occurrence matrix $\mathbf{A}$. This is a variation of the Pointwise Mutual Information (PMI) (Church and Hanks, 1990), in which all PMI values that are less than zero are replaced with zero (Lin, 1998; Bullinaria and Levy, 2007). Let $\mathbf{F}$ be the matrix that results when PPMI is applied to $\mathbf{A}$. Matrix $\mathbf{F}$ has the same number of rows, n_r, and columns, n_c, as the raw co-occurrence matrix $\mathbf{A}$.

Note that in addition to the above-mentioned representation, there are many other ways to represent the distribution of a word in a particular domain (Turney and Pantel, 2010). For example, one can limit the definition of co-occurrence to words that are linked by some dependency relation (Pado and Lapata, 2007), or extend the window of co-occurrence to the entire document (Baroni and Lenci, 2010). Since the method we propose in Section 3.2 to predict the distribution of a word across domains does not depend on the particular

feature representation method, any of these alternative methods could be used.

To reduce the dimensionality of the feature space, and create dense representations for words, we perform SVD on $\mathbf{F}$. We use the left singular vectors corresponding to the k largest singular values to compute a rank k approximation $\hat{\mathbf{F}}$, of $\mathbf{F}$. We perform truncated SVD using SVDLIBC[2]. Each row in $\hat{\mathbf{F}}$ is considered as representing a word in a lower k ($\ll n_c$) dimensional feature space corresponding to a particular domain. Distribution prediction in this lower dimensional feature space is preferrable to prediction over the original feature space because there are reductions in overfitting, feature sparseness, and the learning time. We created two matrices, $\hat{\mathbf{F}}_{\mathcal{S}}$ and $\hat{\mathbf{F}}_{\mathcal{T}}$ from the source and target domains, respectively, using the above mentioned procedure.

3.2 Cross-Domain Feature Vector Prediction

We propose a method to learn a model that can predict the distribution $\boldsymbol{w}_{\mathcal{T}}$ of a word w in the target domain $\mathcal{T}$, given its distribution $\boldsymbol{w}_{\mathcal{S}}$ in the source domain $\mathcal{S}$. We denote the set of features that occur in both domains by $\mathcal{W} = \{w^{(1)}, \ldots, w^{(n)}\}$. In the literature, such features are often referred to as *pivots*, and they have been shown to be useful for DA, allowing the weights learnt to be transferred from one domain to another. Various criteria have been proposed for selecting a small set of pivots for DA, such as the mutual information of a word with the two domains (Blitzer et al., 2007). However, we do not impose any further restrictions on the set of pivots $\mathcal{W}$ other than that they occur in both domains.

For each word $w^{(i)} \in \mathcal{W}$, we denote the corresponding rows in $\hat{\mathbf{F}}_{\mathcal{S}}$ and $\hat{\mathbf{F}}_{\mathcal{T}}$ by column vectors $\boldsymbol{w}_{\mathcal{S}}^{(i)}$ and $\boldsymbol{w}_{\mathcal{T}}^{(i)}$. Note that the dimensionality of $\boldsymbol{w}_{\mathcal{S}}^{(i)}$ and $\boldsymbol{w}_{\mathcal{T}}^{(i)}$ need not be equal, and we may select different numbers of singular vectors to approximate $\hat{\mathbf{F}}_{\mathcal{S}}$ and $\hat{\mathbf{F}}_{\mathcal{T}}$. We model distribution prediction as a multivariate regression problem where, given a set $\{(\boldsymbol{w}_{\mathcal{S}}^{(i)}, \boldsymbol{w}_{\mathcal{T}}^{(i)})\}_{i=1}^{n}$ consisting of pairs of feature vectors selected from each domain for the pivots in $\mathcal{W}$, we learn a mapping from the inputs ($\boldsymbol{w}_{\mathcal{S}}^{(i)}$) to the outputs ($\boldsymbol{w}_{\mathcal{T}}^{(i)}$).

We use Partial Least Squares Regression (PLSR) (Wold, 1985) to learn a regression model using pairs of vectors. PLSR has been applied in Chemometrics (Geladi and Kowalski, 1986), producing stable prediction models even when the number of samples is considerably smaller than the dimensionality of the feature space. In particular, PLSR fits a smaller number of latent variables ($10 - 100$ in practice) such that the correlation between the feature vectors for pivots in the two domains are maximised in this latent space.

Algorithm 1 Learning a prediction model.

Input: $\mathbf{X}$, $\mathbf{Y}$, L.
Output: Prediction matrix $\mathbf{M}$.

1: Randomly select $\boldsymbol{\gamma}_l$ from columns in $\mathbf{Y}_l$.
2: $\boldsymbol{v}_l = {\mathbf{X}_l}^{\top}\boldsymbol{\gamma}_l / \left|\left|{\mathbf{X}_l}^{\top}\boldsymbol{\gamma}_l\right|\right|$
3: $\boldsymbol{\lambda}_l = \mathbf{X}_l\boldsymbol{v}_l$
4: $\boldsymbol{q}_l = {\mathbf{Y}_l}^{\top}\boldsymbol{\lambda}_l / \left|\left|{\mathbf{Y}_l}^{\top}\boldsymbol{\lambda}_l\right|\right|$
5: $\boldsymbol{\gamma}_l = \mathbf{Y}_l\boldsymbol{q}_l$
6: If $\boldsymbol{\gamma}_l$ is unchanged go to Line 7; otherwise go to Line 2
7: $c_l = {\boldsymbol{\lambda}_l}^{\top}\boldsymbol{\gamma}_l / \left|\left|{\boldsymbol{\lambda}_l}^{\top}\boldsymbol{\gamma}_l\right|\right|$
8: $\boldsymbol{p}_l = {\mathbf{X}_l}^{\top}\boldsymbol{\lambda}_l / {\boldsymbol{\lambda}_l}^{\top}\boldsymbol{\lambda}_l$
9: $\mathbf{X}_{l+1} = \mathbf{X}_l - \boldsymbol{\lambda}_l{\boldsymbol{p}_l}^{\top}$ and $\mathbf{Y}_{l+1} = \mathbf{Y}_l - c_l\boldsymbol{\lambda}_l{\boldsymbol{q}_l}^{\top}$.
10: Stop if $l = L$; otherwise $l = l + 1$ and return to Line 1.
11: Let $\mathbf{C} = \mathrm{diag}(c_1, \ldots, c_L)$, and $\mathbf{V} = [\boldsymbol{v}_1 \ldots \boldsymbol{v}_L]$
12: $\mathbf{M} = \mathbf{V}(\mathbf{P}^{\top}\mathbf{V})^{-1}\mathbf{C}\mathbf{Q}^{\top}$
13: **return** $\mathbf{M}$

Let $\mathbf{X}$ and $\mathbf{Y}$ denote matrices formed by arranging respectively the vectors $\boldsymbol{w}_{\mathcal{S}}^{(i)}$s and $\boldsymbol{w}_{\mathcal{T}}^{(i)}$ in rows. PLSR decomposes $\mathbf{X}$ and $\mathbf{Y}$ into a series of products between rank 1 matrices as follows:

$$\mathbf{X} \approx \sum_{l=1}^{L} \boldsymbol{\lambda}_l{\boldsymbol{p}_l}^{\top} = \boldsymbol{\Lambda}\mathbf{P}^{\top} \quad (1)$$

$$\mathbf{Y} \approx \sum_{l=1}^{L} \boldsymbol{\gamma}_l{\boldsymbol{q}_l}^{\top} = \boldsymbol{\Gamma}\mathbf{Q}^{\top}. \quad (2)$$

Here, $\boldsymbol{\lambda}_l$, $\boldsymbol{\gamma}_l$, $\boldsymbol{p}_l$, and $\boldsymbol{q}_l$ are column vectors, and the summation is taken over the rank 1 matrices that result from the outer product of those vectors. The matrices, $\boldsymbol{\Lambda}$, $\boldsymbol{\Gamma}$, $\mathbf{P}$, and $\mathbf{Q}$ are constructed respectively by arranging $\boldsymbol{\lambda}_l$, $\boldsymbol{\gamma}_l$, $\boldsymbol{p}_l$, and $\boldsymbol{q}_l$ vectors as columns.

Our method for learning a distribution prediction model is shown in Algorithm 1. It is based on the two block NIPALS routine (Wold, 1975; Rosipal and Kramer, 2006) and iteratively discovers L pairs of vectors $(\boldsymbol{\lambda}_l, \boldsymbol{\gamma}_l)$ such that the covariances, $\mathrm{Cov}(\boldsymbol{\lambda}_l, \boldsymbol{\gamma}_l)$, are maximised under the constraint $||\boldsymbol{p}_l|| = ||\boldsymbol{q}_l|| = 1$. Finally, the prediction matrix, $\mathbf{M}$ is computed using $\boldsymbol{\lambda}_l, \boldsymbol{\gamma}_l, \boldsymbol{p}_l, \boldsymbol{q}_l$. The predicted distribution $\hat{\boldsymbol{w}}_{\mathcal{T}}$ of a word w in $\mathcal{T}$ is given by

$$\hat{\boldsymbol{w}}_{\mathcal{T}} = \mathbf{M}\boldsymbol{w}_{\mathcal{S}}. \quad (3)$$

[2] http://tedlab.mit.edu/~dr/SVDLIBC/

Our distribution prediction learning method is unsupervised in the sense that it does not require manually labeled data for a particular task from any of the domains. This is an important point, and means that the distribution prediction method is independent of the task to which it may subsequently be applied. As we go on to show in Section 6, this enables us to use the same distribution prediction method for both POS tagging and sentiment classification.

4 Domain Adaptation

The main reason that a model trained only on the source domain labeled data performs poorly in the target domain is the *feature mismatch* – few features in target domain test instances appear in source domain training instances. To overcome this problem, we use the proposed distribution prediction method to find those related features in the source domain that correspond to the features appearing in the target domain test instances.

We consider two DA tasks: (a) cross-domain POS tagging (Section 4.1), and (b) cross-domain sentiment classification (Section 4.2). Note that our proposed distribution prediction method can be applied to numerous other NLP tasks that involve sequence labelling and document classification.

4.1 Cross-Domain POS Tagging

We represent each word using a set of features such as capitalisation (whether the first letter of the word is capitalised), numeric (whether the word contains digits), prefixes up to four letters, and suffixes up to four letters (Miller et al., 2011). Next, for each word w in a source domain labeled (i.e. manually POS tagged) sentence, we select its neighbours $u^{(i)}$ in the source domain as additional features. Specifically, we measure the similarity, $\mathrm{sim}(\boldsymbol{u}_{\mathcal{S}}^{(i)}, \boldsymbol{w}_{\mathcal{S}})$, between the source domain distributions of $u^{(i)}$ and w, and select the top r similar neighbours $u^{(i)}$ for each word w as additional features for w. We refer to such features as *distributional features* in this work. The value of a neighbour $u^{(i)}$ selected as a distributional feature is set to its similarity score $\mathrm{sim}(\boldsymbol{u}_{\mathcal{S}}^{(i)}, \boldsymbol{w}_{\mathcal{S}})$. Next, we train a CRF model using all features (i.e. capitalisation, numeric, prefixes, suffixes, and distributional features) on source domain labeled sentences.

We train a PLSR model, $\mathbf{M}$, that predicts the target domain distribution $\mathbf{M}\boldsymbol{u}_{\mathcal{S}}^{(i)}$ of a word $u^{(i)}$ in the source domain labeled sentences, given its distribution, $\boldsymbol{u}_{\mathcal{S}}^{(i)}$. At test time, for each word w that appears in a target domain test sentence, we measure the similarity, $\mathrm{sim}(\mathbf{M}\boldsymbol{u}_{\mathcal{S}}^{(i)}, \boldsymbol{w}_{\mathcal{T}})$, and select the most similar r words $u^{(i)}$ in the source domain labeled sentences as the distributional features for w, with their values set to $\mathrm{sim}(\mathbf{M}\boldsymbol{u}_{\mathcal{S}}^{(i)}, \boldsymbol{w}_{\mathcal{T}})$. Finally, the trained CRF model is applied to a target domain test sentence.

Note that distributional features are always selected from the source domain during both train and test times, thereby increasing the number of overlapping features between the trained model and test sentences. To make the inference tractable and efficient, we use a first-order Markov factorisation, in which we consider all pairwise combinations between the features for the current word and its immediate predecessor.

4.2 Cross-Domain Sentiment Classification

Unlike in POS tagging, where we must individually tag each word in a target domain test sentence, in sentiment classification we must classify the sentiment for the entire review. We modify the DA method presented in Section 4.1 to satisfy this requirement as follows.

Let us assume that we are given a set $\{(\boldsymbol{x}_{\mathcal{S}}^{(i)}, y^{(i)})\}_{i=1}^{n}$ of n labeled reviews $\boldsymbol{x}_{\mathcal{S}}^{(i)}$ for the source domain $\mathcal{S}$. For simplicity, let us consider binary sentiment classification where each review $\boldsymbol{x}^{(i)}$ is labeled either as positive (i.e. $y^{(i)} = 1$) or negative (i.e. $y^{(i)} = -1$). Our cross-domain binary sentiment classification method can be easily extended to the multi-class setting as well. First, we lemmatise each word in a source domain labeled review $\boldsymbol{x}_{\mathcal{S}}^{(i)}$, and extract both unigrams and bigrams as features to represent $\boldsymbol{x}_{\mathcal{S}}^{(i)}$ by a binary-valued feature vector. Next, we train a binary classification model, $\boldsymbol{\theta}$, using those feature vectors. Any binary classification algorithm can be used to learn $\boldsymbol{\theta}$. In our experiments, we used L2 regularised logistic regression.

Next, we train a PLSR model, $\mathbf{M}$, as described in Section 3.2 using unlabeled reviews in the source and target domains. At test time, we represent a test target review H using a binary-valued feature vector $\boldsymbol{h}$ of unigrams and bigrams of lemmas of the words in H, as we did for source domain labeled train reviews. Next, for each feature $w^{(j)}$ extracted from H, we measure the similarity,

$\text{sim}(\mathbf{M}\boldsymbol{u}_{\mathcal{S}}^{(i)}, \boldsymbol{w}_{\mathcal{T}}^{(j)})$, between the target domain distribution of $w^{(j)}$, and each feature (unigram or bigram) $u^{(i)}$ in the source domain labeled reviews. We score each source domain feature $u^{(i)}$ for its relatedness to H using the formula:

$$\text{score}(u^{(i)}, \mathsf{H}) = \frac{1}{|\mathsf{H}|} \sum_{j=1}^{|\mathsf{H}|} \text{sim}(\mathbf{M}\boldsymbol{u}_{\mathcal{S}}^{(i)}, \boldsymbol{w}_{\mathcal{T}}^{(j)}) \quad (4)$$

where $|\mathsf{H}|$ denotes the total number of features extracted from the test review H. We select the top scoring r features $u^{(i)}$ as distributional features for H, and append those to $\boldsymbol{h}$. The corresponding values of those distributional features are set to the scores given by Equation 4. Finally, we classify $\boldsymbol{h}$ using the trained binary classifier $\boldsymbol{\theta}$. Note that given a test review, we find the distributional features that are similar to *all* the words in the test review from the source domain. In particular, we *do not* find distributional features independently for each word in the test review. This enables us to find distributional features that are consistent with all the features in a test review.

4.3 Model Choices

For both POS tagging and sentiment classification, we experimented with several alternative approaches for feature weighting, representation, and similarity measures using development data, which we randomly selected from the training instances from the datasets described in Section 5.

For feature weighting for sentiment classification, we considered using the number of occurrences of a feature in a review and tf-idf weighting (Salton and Buckley, 1983). For representation, we considered distributional features $u^{(i)}$ in descending order of their scores given by Equation 4, and then taking the inverse-rank as the values for the distributional features (Bollegala et al., 2011). However, none of these alternatives resulted in performance gains. With respect to similarity measures, we experimented with cosine similarity and the similarity measure proposed by Lin (1998); cosine similarity performed consistently well over all the experimental settings. The feature representation was held fixed during these similarity measure comparisons.

For POS tagging, we measured the effect of varying r, the number of distributional features, using a development dataset. We observed that setting r larger than 10 did not result in significant improvements in tagging accuracy, but only increased the train time due to the larger feature space. Consequently, we set $r = 10$ in POS tagging. For sentiment analysis, we used all features in the source domain labeled reviews as distributional features, weighted by their scores given by Equation 4, taking the inverse-rank. In both tasks, we parallelised similarity computations using BLAS[3] level-3 routines to speed up the computations. The source code of our implementation is publicly available[4].

5 Datasets

To evaluate DA for POS tagging, following Blitzer et al. (2006), we use sections $2 - 21$ from Wall Street Journal (WSJ) as the source domain labeled data. An additional $100,000$ WSJ sentences from the 1988 release of the WSJ corpus are used as the source domain unlabeled data. Following Schnabel and Schütze (2013), we use the POS labeled sentences in the SACNL dataset (Petrov and McDonald, 2012) for the five target domains: QA forums, Emails, Newsgroups, Reviews, and Blogs. Each target domain contains around 1000 POS labeled test sentences and around $100,000$ unlabeled sentences.

To evaluate DA for sentiment classification, we use the Amazon product reviews collected by Blitzer et al. (2007) for four different product categories: books (**B**), DVDs (**D**), electronic items (**E**), and kitchen appliances (**K**). There are 1000 positive and 1000 negative sentiment labeled reviews for each domain. Moreover, each domain has on average $17,547$ unlabeled reviews. We use the standard split of 800 positive and 800 negative labeled reviews from each domain as training data, and the remainder for testing.

6 Experiments and Results

For each domain $\mathcal{D}$ in the SANCL (POS tagging) and Amazon review (sentiment classification) datasets, we create a PPMI weighted co-occurrence matrix $\mathbf{F}_{\mathcal{D}}$. On average, $\mathbf{F}_{\mathcal{D}}$ created for a target domain in the SANCL dataset contains $104,598$ rows and $65,528$ columns, whereas those numbers in the Amazon dataset are $27,397$ and $35,200$ respectively. In cross-domain sentiment classification, we measure the binary sentiment classification accuracy for the target domain

[3] http://www.openblas.net/

[4] http://www.csc.liv.ac.uk/~danushka/software.html

test reviews for each pair of domains (12 pairs in total for 4 domains). On average, we have $40,176$ pivots for a pair of domains in the Amazon dataset.

In cross-domain POS tagging, WSJ is always the source domain, whereas the five domains in SANCL dataset are considered as the target domains. For this setting we have 9822 pivots on average. The number of singular vectors k selected in SVD, and the number of PLSR dimensions L are set respectively to 1000 and 50 for the remainder of the experiments described in the paper. Later we study the effect of those two parameters on the performance of the proposed method. The L-BFGS (Liu and Nocedal, 1989) method is used to train the CRF and logistic regression models.

6.1 POS Tagging Results

Table 2 shows the token-level POS tagging accuracy for *unseen* words (i.e. words that appear in the target domain test sentences but not in the source domain labeled train sentences). By limiting the evaluation to unseen words instead of all words, we can evaluate the gain in POS tagging accuracy solely due to DA. The **NA** (no-adapt) baseline simulates the effect of not performing any DA. Specifically, in POS tagging, a CRF trained on source domain labeled sentences is applied to target domain test sentences, whereas in sentiment classification, a logistic regression classifier trained using source domain labeled reviews is applied to the target domain test reviews. The $\mathcal{S}_{\mathbf{pred}}$ baseline directly uses the source domain distributions for the words instead of projecting them to the target domain. This is equivalent to setting the prediction matrix $\mathbf{M}$ to the unit matrix. The $\mathcal{T}_{\mathbf{pred}}$ baseline uses the target domain distribution $\boldsymbol{w}_{\mathcal{T}}$ for a word w instead of $\mathbf{M}\boldsymbol{w}_{\mathcal{S}}$. If w does not appear in the target domain, then $\boldsymbol{w}_{\mathcal{T}}$ is set to the zero vector. The $\mathcal{S}_{pred}$ and $\mathcal{T}_{pred}$ baselines simulate the two alternatives of using source and target domain distributions instead of learning a PLSR model. The DA method proposed in Section 4.1 is shown as the **Proposed** method. **Filter** denotes the training set filtering method proposed by Schnabel and Schütze (2013) for the DA of POS taggers.

From Table 2, we see that the **Proposed** method achieves the best performance in all five domains, followed by the $\mathcal{T}_{pred}$ baseline. Recall that the $\mathcal{T}_{pred}$ baseline cannot find source domain words that do not appear in the target domain as distributional features for the words in the target domain test reviews. Therefore, when the overlap between the vocabularies used in the source and the target domains is small, $\mathcal{T}_{pred}$ cannot reduce the mismatch between the feature spaces. Poor performance of the $\mathcal{S}_{pred}$ baseline shows that the distributions of a word in the source and target domains are different to the extent that the distributional features found using source domain distributions are inadequate. The two baselines $\mathcal{S}_{pred}$ and $\mathcal{T}_{pred}$ collectively motivate our proposal to learn a distribution prediction model from the source domain to the target. The improvements of **Proposed** over the previously proposed **Filter** are statistically significant in all domains except the Emails domain (denoted by † in Table 2 according to the Binomial exact test at 95% confidence). However, the differences between the $\mathcal{T}_{pred}$ and **Proposed** methods are not statistically significant.

Target	NA	$\mathcal{S}_{pred}$	$\mathcal{T}_{pred}$	Filter	Proposed
QA	67.34	68.18	68.75	57.08	$\mathbf{69.28}^{\dagger}$
Emails	65.62	66.62	67.07	65.61	**67.09**
Newsgroups	75.71	75.09	75.57	70.37	$\mathbf{75.85}^{\dagger}$
Reviews	56.36	54.60	56.68	47.91	$\mathbf{56.93}^{\dagger}$
Blogs	76.64	54.78	76.90	74.56	$\mathbf{76.97}^{\dagger}$

Table 2: POS tagging accuracies on SANCL.

6.2 Sentiment Classification Results

In Figure 1, we compare the **Proposed** cross-domain sentiment classification method (Section 4.2) against several baselines and the current state-of-the-art methods. The baselines **NA**, $\mathcal{S}_{\mathbf{pred}}$, and $\mathcal{T}_{\mathbf{pred}}$ are defined similarly as in Section 6.1. **SST** is the Sentiment Sensitive Thesaurus proposed by Bollegala et al. (2011). **SST** creates a single distribution for a word using both source and target domain reviews, instead of two separate distributions as done by the **Proposed** method. **SCL** denotes the Structural Correspondence Learning method proposed by Blitzer et al. (2006). **SFA** denotes the Spectral Feature Alignment method proposed by Pan et al. (2010). **SFA** and **SCL** represent the current state-of-the-art methods for cross-domain sentiment classification. All methods are evaluated under the same settings, including train/test split, feature spaces, pivots, and classification algorithms so that any differences in performance can be directly attributable to their domain adaptability. For each domain, the accuracy obtained by a classifier trained using labeled data from that

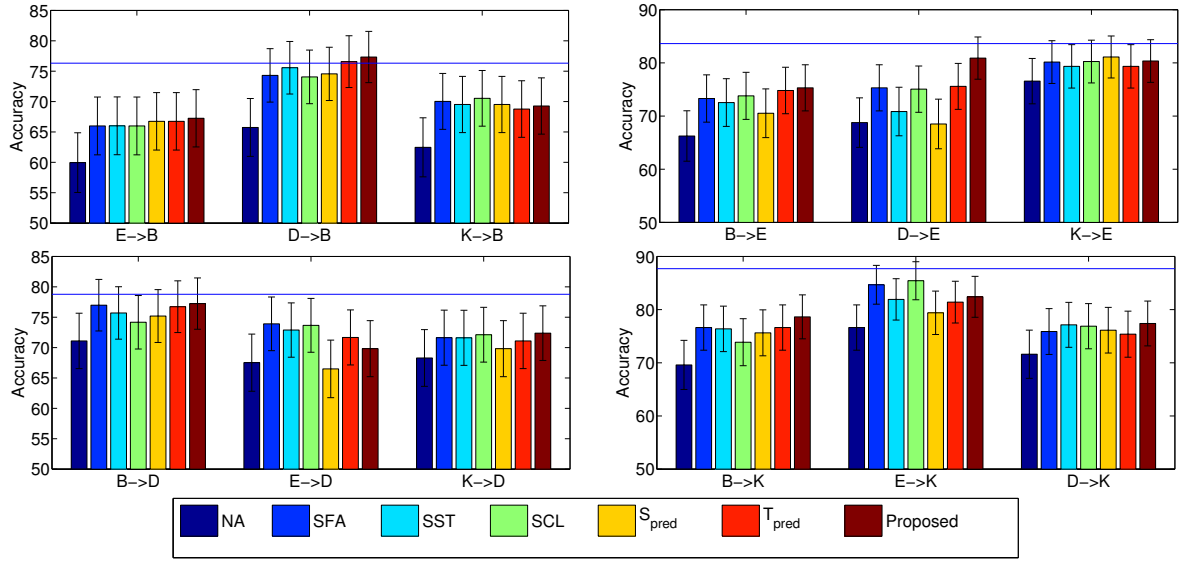


Figure 1: Cross-Domain sentiment classification.

domain is indicated by a solid horizontal line in each sub-figure. This upper baseline represents the classification accuracy we could hope to obtain if we were to have labeled data for the target domain. Clopper-Pearson 95% binomial confidence intervals are superimposed on each vertical bar.

From Figure 1 we see that the **Proposed** method reports the best results in 8 out of the 12 domain pairs, whereas **SCL**, **SFA**, and $\mathcal{S}_{\mathbf{pred}}$ report the best results in other cases. Except for the **D-E** setting in which **Proposed** method significantly outperforms both **SFA** and **SCL**, the performance of the **Proposed** method is not statistically significantly different to that of **SFA** or **SCL**.

The selection of pivots is vital to the performance of **SFA**. However, unlike **SFA**, which requires us to carefully select a small subset of pivots (ca. less than 500) using some heuristic approach, our **Proposed** method does not require any pivot selection. Moreover, **SFA** projects source domain reviews to a lower-dimensional latent space, in which a binary sentiment classifier is subsequently trained. At test time **SFA** projects a target review into this lower-dimensional latent space and applies the trained classifier. In contrast, our **Proposed** method predicts the distribution of a word in the target domain, given its distribution in the source domain, thereby explicitly *translating* the source domain reviews to the target. This property enables us to apply the proposed distribution prediction method to tasks other than sentiment analysis such as POS tagging where we must identify distributional features for individual words.

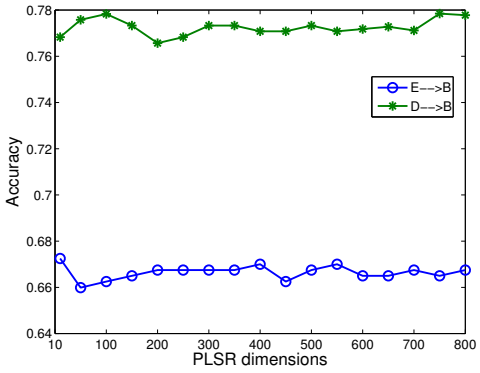


Figure 2: The effect of PLSR dimensions.

Unlike our distribution prediction method, which is unsupervised, **SST** requires labeled data for the source domain to learn a feature mapping between a source and a target domain in the form of a thesaurus. However, from Figure 1 we see that in 10 out of the 12 domain-pairs the **Proposed** method returns higher accuracies than **SST**.

To evaluate the overall effect of the number of singular vectors k used in the SVD step, and the number of PLSR components L used in Algorithm 1, we conduct two experiments. To evaluate the effect of the PLSR dimensions, we fixed $k = 1000$ and measured the cross-domain sentiment classification accuracy over a range of L values. As shown in Figure 2, accuracy remains stable across a wide range of PLSR dimensions. Because the time complexity of Algorithm 1 increases linearly with L, it is desirable that we select smaller L val-

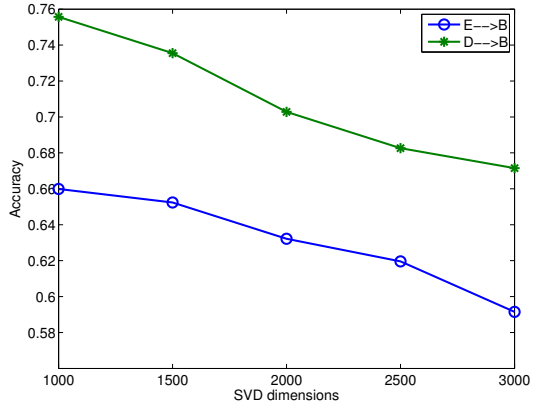


Figure 3: The effect of SVD dimensions.

Measure	Distributional features
$\mathrm{sim}(u_{\mathcal{S}}, w_{\mathcal{S}})$	thin (0.1733), digestible (0.1728), small+print (0.1722)
$\mathrm{sim}(u_{\mathcal{T}}, w_{\mathcal{T}})$	travel+companion (0.6018), snap-in (0.6010), touchpad (0.6016)
$\mathrm{sim}(u_{\mathcal{S}}, w_{\mathcal{T}})$	segregation (0.1538), participation (0.1512), depression+era (0.1508)
$\mathrm{sim}(\mathbf{M}u_{\mathcal{S}}, w_{\mathcal{T}})$	small (0.2794), compact (0.2641), sturdy (0.2561)

Table 3: Top 3 distributional features $u \in \mathcal{S}$ for the word *lightweight* (w).

ues in practice.

To evaluate the effect of the SVD dimensions, we fixed $L = 100$ and measured the cross-domain sentiment classification accuracy for different k values as shown in Figure 3. We see an overall decrease in classification accuracy when k is increased. Because the dimensionality of the source and target domain feature spaces is equal to k, the complexity of the least square regression problem increases with k. Therefore, larger k values result in overfitting to the train data and classification accuracy is reduced on the target test data.

As an example of the distribution prediction method, in Table 3 we show the top 3 similar distributional features u in the *books* (source) domain, predicted for the *electronics* (target) domain word w = *lightweight*, by different similarity measures. Bigrams are indicted by a + sign and the similarity scores of the distributional features are shown within brackets.

Using the source domain distributions for both u and w (i.e. $\mathrm{sim}(u_{\mathcal{S}}, w_{\mathcal{S}})$) produces distributional features that are specific to the books domain, or to the dominant adjectival sense of *having no importance or influence*. On the other hand, using target domain distributions for u and w (i.e. $\mathrm{sim}(u_{\mathcal{T}}, w_{\mathcal{T}})$) returns distributional features of the dominant nominal sense of *lower in weight* frequently associated with electronic devices. Simply using source domain distributions $u_{\mathcal{S}}$ (i.e. $\mathrm{sim}(u_{\mathcal{S}}, w_{\mathcal{T}})$) returns totally unrelated distributional features. This shows that word distributions in source and target domains are very different and some adaptation is required prior to computing distributional features.

Interestingly, we see that by using the distributions predicted by the proposed method (i.e. $\mathrm{sim}(\mathbf{M}u_{\mathcal{S}}, w_{\mathcal{T}})$) we overcome this problem and find relevant distributional features from the source domain. Although for illustrative purposes we used the word *lightweight*, which occurs in both the source and the target domains, our proposed method does not require the source domain distribution $w_{\mathcal{S}}$ for a word w in a target domain document. Therefore, it can find distributional features even for words occurring only in the target domain, thereby reducing the feature mismatch between the two domains.

7 Conclusion

We proposed a method to predict the distribution of a word across domains. We first create a distributional representation for a word using the data from a single domain, and then learn a Partial Least Square Regression (PLSR) model to predict the distribution of a word in a target domain given its distribution in a source domain. We evaluated the proposed method in two domain adaptation tasks: cross-domain POS tagging and cross-domain sentiment classification. Our experiments show that without requiring any task-specific customisations to our distribution prediction method, it outperforms competitive baselines and achieves comparable results to the current state-of-the-art domain adaptation methods.

References

Anthony Aue and Michael Gamon. 2005. Customizing sentiment classifiers to new domains: a case study. Technical report, Microsoft Research.

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

Romaric Besançon, Martin Rajman, and Jean-Cédric Chappelier. 1999. Textual similarities based on a

distributional approach. In *Proc. of DEXA*, pages 180 – 184.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proc. of EMNLP*, pages 120 – 128.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proc. of ACL*, pages 440 – 447.

Danushka Bollegala, David Weir, and John Carroll. 2011. Using multiple sources to construct a sentiment sensitive thesaurus for cross-domain sentiment classification. In *Proc. of ACL/HLT*, pages 132 – 141.

Ted Briscoe, John Carroll, and Rebecca Watson. 2006. The second release of the RASP system. In *Proc. of COLING/ACL Interactive Presentation Sessions*.

John A. Bullinaria and Jospeh P. Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, 39(3):510 – 526.

Jinho D. Choi and Martha Palmer. 2012. Fast and robust part-of-speech tagging using dynamic model selection. In *Proc. of ACL Short Papers*, volume 2, pages 363 – 367.

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

James Curran. 2005. Supersense tagging of unknown nouns using semantic similarity. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 26 – 33.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proc. of ACL*, pages 256 – 263.

John R. Firth. 1957. A synopsis of linguistic theory 1930-55. *Studies in Linguistic Analysis*, pages 1 – 32.

Paul Geladi and Bruce R. Kowalski. 1986. Partial least-squares regression: a tutorial. *Analytica Chimica Acta*, 185(0):1 – 17.

Honglei Guo, Huijia Zhu, Zhili Guo, Xiaoxun Zhang, Xian Wu, and Zhong Su. 2009. Domain adaptation with latent semantic association for named entity recognition. In *Proc. of NAACL*, pages 281 – 289.

Yulan He, Chenghua Lin, and Harith Alani. 2011. Automatically extracting polarity-bearing topics for cross-domain sentiment classification. In *Proc. of ACL/HLT*, pages 123 – 131.

Fei Huang and Alexander Yates. 2009. Distributional representations for handling sparsity in supervised sequence-labeling. In *ACL-IJCNLP'09*, pages 495 – 503.

Fei Huang and Alexander Yates. 2012. Biased representation learning for domain adaptation. In *Proc. of EMNLP/CoNLL*, pages 1313 – 1323.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2012. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16(4):359 – 389.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proc. of ACL*, pages 768 – 774.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45:503 – 528.

David McClosky, Eugene Charniak, and Mark Johnson. 2010. Automatic domain adaptation for parsing. In *Proc. of NAACL/HLT*, pages 28 – 36.

John E. Miller, Manabu Torii, and K. Vijay-Shanker. 2011. Building domain-specific taggers without annotated (domain) data. In *Proc. of EMNLP/CoNLL*, pages 1103 – 1111.

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

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *Proc. of WWW*, pages 751 – 760.

Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. 2009. Web-scale distributional similarity and entity set expansion. In *Proc. of EMNLP*, pages 938 – 947.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *Notes of the 1st SANCL Workshop*.

Natalia Ponomareva and Mike Thelwall. 2012. Do neighbours help? an exploration of graph-based algorithms for cross-domain sentiment classification. In *Proc. of EMNLP*, pages 655 – 665.

Roman Rosipal and Nicole Kramer. 2006. Overview and recent advances in partial least squares. In C. Saunders et al., editor, *SLSFS'05*, volume 3940 of *LNCS*, pages 34 – 51, Berlin Heidelberg. Springer-Verlag.

G. Salton and C. Buckley. 1983. *Introduction to Modern Information Retreival*. McGraw-Hill Book Company.

Tobias Schnabel and Hinrich Schütze. 2013. Towards robust cross-domain domain adaptation for part-of-speech tagging. In *Proc. of IJCNLP*, pages 198 – 206.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of EMNLP*, pages 1631 – 1642.

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

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antonie Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371 – 3408.

Herman Wold. 1975. Path models with latent variables: the NIPALS approach. In H. M. Blalock et al., editor, *Quantitative socialogy: international perspective on mathematical and statistical modeling*, pages 307 – 357. Academic.

Herman Wold. 1985. Partial least squares. In Samel Kotz and Norman L. Johnson, editors, *Encyclopedia of the Statistical Sciences*, pages 581 – 591. Wiley.

Min Xiao, Feipeng Zhao, and Yuhong Guo. 2013. Learning latent word representations for domain adaptation using supervised word clustering. In *Proc. of EMNLP*, pages 152 – 162.

How to make words with vectors: Phrase generation in distributional semantics

Georgiana Dinu and **Marco Baroni**
Center for Mind/Brain Sciences
University of Trento, Italy
(georgiana.dinu|marco.baroni)@unitn.it

Abstract

We introduce the problem of *generation* in distributional semantics: Given a distributional vector representing some meaning, how can we generate the phrase that best expresses that meaning? We motivate this novel challenge on theoretical and practical grounds and propose a simple data-driven approach to the estimation of generation functions. We test this in a monolingual scenario (paraphrase generation) as well as in a cross-lingual setting (translation by synthesizing adjective-noun phrase vectors in English and generating the equivalent expressions in Italian).

1 Introduction

Distributional methods for semantics approximate the meaning of linguistic expressions with vectors that summarize the contexts in which they occur in large samples of text. This has been a very successful approach to lexical semantics (Erk, 2012), where semantic relatedness is assessed by comparing vectors. Recently these methods have been extended to phrases and sentences by means of composition operations (see Baroni (2013) for an overview). For example, given the vectors representing *red* and *car*, composition derives a vector that approximates the meaning of *red car*.

However, the link between language and meaning is, obviously, bidirectional: As message recipients we are exposed to a linguistic expression and we must compute its meaning (the *synthesis* problem). As message producers we start from the meaning we want to communicate (a "thought") and we must encode it into a word sequence (the *generation* problem). If distributional semantics is to be considered a proper semantic theory, then it must deal not only with synthesis (going from words to vectors), but also with generation (from vectors to words).

Besides these theoretical considerations, phrase generation from vectors has many useful applications. We can, for example, synthesize the vector representing the meaning of a phrase or sentence, and then generate alternative phrases or sentences from this vector to accomplish true *paraphrase generation* (as opposed to paraphrase detection or ranking of candidate paraphrases).

Generation can be even more useful when the source vector comes from another modality or language. Recent work on grounding language in vision shows that it is possible to represent images and linguistic expressions in a common vector-based semantic space (Frome et al., 2013; Socher et al., 2013). Given a vector representing an image, generation can be used to productively construct phrases or sentences that describe the image (as opposed to simply retrieving an existing description from a set of candidates). Translation is another potential application of the generation framework: Given a semantic space shared between two or more languages, one can compose a word sequence in one language and generate translations in another, with the shared semantic vector space functioning as interlingua.

Distributional semantics assumes a lexicon of atomic expressions (that, for simplicity, we take to be *words*), each associated to a vector. Thus, at the single-word level, the problem of generation is solved by a trivial *generation-by-synthesis* approach: Given an arbitrary target vector, "generate" the corresponding word by searching through the lexicon for the word with the closest vector to the target. This is however unfeasible for larger expressions: Given n vocabulary elements, this approach requires checking n^k phrases of length k. This becomes prohibitive already for relatively short phrases, as reasonably-sized vocabularies do not go below tens of thousands of words. The search space for 3-word phrases in a 10K-word vocabulary is already in the order of trillions. In

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 624–633,
Baltimore, Maryland, USA, June 23-25 2014.

this paper, we introduce a more direct approach to phrase generation, inspired by the work in compositional distributional semantics. In short, we revert the composition process and we propose a framework of data-induced, syntax-dependent functions that *decompose* a single vector into a vector sequence. The generated vectors can then be efficiently matched against those in the lexicon or fed to the decomposition system again to produce longer phrases recursively.

2 Related work

To the best of our knowledge, we are the first to explicitly and systematically pursue the generation problem in distributional semantics. Kalchbrenner and Blunsom (2013) use top-level, composed distributed representations of sentences to guide generation in a machine translation setting. More precisely, they condition the target language model on the composed representation (addition of word vectors) of the source language sentence.

Andreas and Ghahramani (2013) discuss the the issue of generating language from vectors and present a probabilistic generative model for distributional vectors. However, their emphasis is on *reversing* the generative story in order to derive composed meaning representations from word sequences. The theoretical generating capabilities of the methods they propose are briefly exemplified, but not fully explored or tested.

Socher et al. (2011) come closest to our target problem. They introduce a bidirectional language-to-meaning model for compositional distributional semantics that is similar in spirit to ours. However, we present a clearer decoupling of synthesis and generation and we use different (and simpler) training methods and objective functions. Moreover, Socher and colleagues do not train separate decomposition rules for different syntactic configurations, so it is not clear how they would be able to control the generation of different output structures. Finally, the potential for generation is only addressed in passing, by presenting a few cases where the generated sequence has the same syntactic structure of the input sequence.

3 General framework

We start by presenting the familiar synthesis setting, focusing on two-word phrases. We then introduce generation for the same structures. Finally, we show how synthesis and generation of longer phrases is handled by recursive extension of the two-word case. We assume a lexicon L, that is, a bi-directional look-up table containing a list of words L_w linked to a matrix L_v of vectors. Both synthesis and generation involve a trivial lexicon look-up step to retrieve vectors associated to words and *vice versa*: We ignore it in the exposition below.

3.1 Synthesis

To construct the vector representing a two-word phrase, we must compose the vectors associated to the input words. More formally, similarly to Mitchell and Lapata (2008), we define a syntax-dependent *composition* function yielding a phrase vector $\vec{p}$:

$$\vec{p} = \mathrm{f}_{\mathrm{comp_R}}(\vec{u}, \vec{v})$$

where $\vec{u}$ and $\vec{v}$ are the vector representations associated to words u and v. $\mathrm{f}_{\mathrm{comp_R}} : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ (for d the dimensionality of vectors) is a composition function specific to the syntactic relation R holding between the two words.[1]

Although we are not bound to a specific composition model, throughout this paper we use the method proposed by Guevara (2010) and Zanzotto et al. (2010) which defines composition as application of linear transformations to the two constituents followed by summing the resulting vectors: $\mathrm{f}_{\mathrm{comp_R}}(\vec{u}, \vec{v}) = W_1\vec{u} + W_2\vec{v}$. We will further use the following equivalent formulation:

$$\mathrm{f}_{\mathrm{comp_R}}(\vec{u}, \vec{v}) = W_R[\vec{u}; \vec{v}]$$

where $W_R \in R^{d \times 2d}$ and $[\vec{u}; \vec{v}]$ is the vertical concatenation of the two vectors (using Matlab notation). Following Guevara, we learn W_R using examples of word and phrase vectors directly extracted from the corpus (for the rest of the paper, we refer to these phrase vectors extracted non-compositionally from the corpus as *observed* vectors). To estimate, for example, the weights in the W_{AN} (adjective-noun) matrix, we use the corpus-extracted vectors of the words in tuples such as $\langle red, car, red.car \rangle$, $\langle evil, cat, evil.cat \rangle$, etc. Given a set of training examples stacked into matrices U, V (the constituent vectors) and P (the corresponding observed vectors), we estimate W_R by solving the least-squares regression problem:

[1] Here we make the simplifying assumption that all vectors have the same dimensionality, however this need not necessarily be the case.

$$\min_{W_R \in \mathbb{R}^{d \times 2d}} \|P - W_R[U;V]\| \quad (1)$$

We use the approximation of observed phrase vectors as objective because these vectors can provide direct evidence of the polysemous behaviour of words: For example, the corpus-observed vectors of *green jacket* and *green politician* reflect how the meaning of *green* is affected by its occurrence with different nouns. Moreover, it has been shown that for two-word phrases, despite their relatively low frequency, such corpus-observed representations are still difficult to outperform in phrase similarity tasks (Dinu et al., 2013; Turney, 2012).

3.2 Generation

Generation of a two-word sequence from a vector proceeds in two steps: decomposition of the phrase vectors into two constituent vectors, and search for the nearest neighbours of each constituent vector in L_v (the lexical matrix) in order to retrieve the corresponding words from L_w.

Decomposition We define a syntax-dependent *decomposition* function:

$$[\vec{u};\vec{v}] = \mathrm{f}_{\mathrm{decomp_R}}(\vec{p})$$

where $\vec{p}$ is a phrase vector, $\vec{u}$ and $\vec{v}$ are vectors associated to words standing in the syntactic relation R and $\mathrm{f}_{\mathrm{decomp_R}} : \mathbb{R}^d \rightarrow \mathbb{R}^d \times \mathbb{R}^d$.

We assume that decomposition is also a linear transformation, $W'_R \in \mathbb{R}^{2d \times d}$, which, given an input phrase vector, returns two constituent vectors:

$$\mathrm{f}_{\mathrm{decomp_R}}(\vec{p}) = W'_R\vec{p}$$

Again, we can learn from corpus-observed vectors associated to tuples of word pairs and the corresponding phrases by solving:

$$\min_{W'_R \in \mathbb{R}^{2d \times d}} \|[U;V] - W'_R P\| \quad (2)$$

If a composition function $\mathrm{f}_{\mathrm{comp_R}}$ is available, an alternative is to learn a function that can best *revert* this composition. The decomposition function is then trained as follows:

$$\min_{W'_R \in \mathbb{R}^{2d \times d}} \|[U;V] - W'_R W_R[U;V]\| \quad (3)$$

where the matrix W_R is a given composition function for the same relation R. Training with observed phrases, as in eq. (2), should be better at capturing the idiosyncrasies of the actual distribution of phrases in the corpus and it is more robust by being independent from the availability and quality of composition functions. On the other hand, if the goal is to revert as faithfully as possible the composition process and retrieve the original constituents (e.g., in a different modality or a different language), then the objective in eq. (3) is more motivated.

Nearest neighbour search We retrieve the nearest neighbours of each constituent vector $\vec{u}$ obtained by decomposition by applying a search function s:

$$\mathrm{NN}_{\vec{u}} = s(\vec{u}, L_v, t)$$

where $\mathrm{NN}_{\vec{u}}$ is a list containing the t nearest neighours of $\vec{u}$ from L_v, the lexical vectors. Depending on the task, t might be set to 1 to retrieve just one word sequence, or to larger values to retrieve t alternatives. The similarity measure used to determine the nearest neighbours is another parameter of the search function; we omit it here as we only experiment with the standard cosine measure (Turney and Pantel, 2010).[2]

3.3 Recursive (de)composition

Extension to longer sequences is straightforward if we assume binary tree representations as syntactic structures. In synthesis, the top-level vector can be obtained by applying composition functions recursively. For example, the vector of *big red car* would be obtained as: $\mathrm{f}_{\mathrm{comp_{AN}}}(\vec{big}, \mathrm{f}_{\mathrm{comp_{AN}}}(\vec{red}, \vec{car}))$, where $\mathrm{f}_{\mathrm{comp_{AN}}}$ is the composition function for adjective-noun phrase combinations. Conversely, for generation, we decompose the phrase vector with $\mathrm{f}_{\mathrm{decomp_{AN}}}$. The first vector is used for retrieving the nearest adjective from the lexicon, while the second vector is further decomposed.

In the experiments in this paper we assume that the syntactic structure is given. In Section 7, we discuss ways to eliminate this assumption.

[2]Note that in terms of computational efficiency, cosine-based nearest neighbour searches reduce to vector-matrix multiplications, for which many efficient implementations exist. Methods such as locality sensitive hashing can be used for further speedups when working with particularly large vocabularies (Andoni and Indyk, 2008).

4 Evaluation setting

In our empirical part, we focus on noun phrase generation. A noun phrase can be a single noun or a noun with one or more modifiers, where a modifier can be an adjective or a prepositional phrase. A prepositional phrase is in turn composed of a preposition and a noun phrase. We learn two composition (and corresponding decomposition) functions: one for modifier-noun phrases, trained on adjective-noun (AN) pairs, and a second one for prepositional phrases, trained on preposition-noun (PN) combinations. For the rest of this section we describe the construction of the vector spaces and the (de)composition function learning procedure.

Construction of vector spaces We test two types of vector representations. The *cbow* model introduced in Mikolov et al. (2013a) learns vector representations using a neural network architecture by trying to predict a target word given the words surrounding it. We use the word2vec software[3] to build vectors of size 300 and using a context window of 5 words to either side of the target. We set the sub-sampling option to 1e-05 and estimate the probability of a target word with the negative sampling method, drawing 10 samples from the noise distribution (see Mikolov et al. (2013a) for details). We also implement a standard *count*-based bag-of-words distributional space (Turney and Pantel, 2010) which counts occurrences of a target word with other words within a symmetric window of size 5. We build a 300Kx300K symmetric co-occurrence matrix using the top most frequent words in our source corpus, apply positive PMI weighting and Singular Value Decomposition to reduce the space to 300 dimensions. For both spaces, the vectors are finally normalized to unit length.[4]

For both types of vectors we use 2.8 billion tokens as input (ukWaC + Wikipedia + BNC). The Italian language vectors for the cross-lingual experiments of Section 6 were trained on 1.6 billion tokens from itWaC.[5] A word token is a word-form + POS-tag string. We extract both word vectors and the observed phrase vectors which are required for the training procedures. We sanity-check the two spaces on MEN (Bruni et al., 2012), a 3,000 items word similarity data set. *cbow* significantly outperforms *count* (0.80 vs. 0.72 Spearman correlations with human judgments). *count* performance is consistent with previously reported results.[6]

(De)composition function training The training data sets consist of the 50K most frequent $\langle u, v, p\rangle$ tuples for each phrase type, for example, $\langle red, car, red.car\rangle$ or $\langle in, car, in.car\rangle$.[7] We concatenate $\vec{u}$ and $\vec{v}$ vectors to obtain the $[U; V]$ matrix and we use the observed $\vec{p}$ vectors (e.g., the corpus vector of the *red.car* bigram) to obtain the phrase matrix P. We use these data sets to solve the least squares regression problems in eqs. (1) and (2), obtaining estimates of the composition and decomposition matrices, respectively. For the decomposition function in eq. (3), we replace the observed phrase vectors with those composed with $\mathrm{f}_{\mathrm{comp_R}}(\vec{u}, \vec{v})$, where $\mathrm{f}_{\mathrm{comp_R}}$ is the previously estimated composition function for relation R.

Composition function performance Since the experiments below also use composed vectors as input to the generation process, it is important to provide independent evidence that the composition model is of high quality. This is indeed the case: We tested our composition approach on the task of retrieving *observed* AN and PN vectors, based on their composed vectors (similarly to Baroni and Zamparelli (2010), we want to retrieve the observed *red.car* vector using $\mathrm{f}_{\mathrm{comp_{AN}}}(red, car)$). We obtain excellent results, with minimum accuracy of 0.23 (chance level <0.0001). We also test on the AN-N paraphrasing test set used in Dinu et al. (2013) (in turn adapting Turney (2012)). The dataset contains 620 ANs, each paired with a single-noun paraphrase (e.g., *false belief/fallacy*, *personal appeal/charisma*). The task is to rank all nouns in the lexicon by their similarity to the phrase, and return the rank of the correct paraphrase. Results are reported in the first row of Table 1. To facilitate comparison, we search, like Dinu et al., through a vocabulary containing the 20K most frequent nouns. The *count* vectors results are similar to those reported by Dinu and colleagues for the same model, and with *cbow* vec-

[3] Available at `https://code.google.com/p/word2vec/`

[4] The parameters of both models have been chosen without specific tuning, based on their observed stable performance in previous independent experiments.

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

[6] See Baroni et al. (2014) for an extensive comparison of the two types of vector representations.

[7] For PNs, we ignore determiners and we collapse, for example, *in.the.car* and *in.car* occurrences.

Input	Output	**cbow**	**count**
A∘N	**N**	11	171
N	**A, N**	67,29	204,168

Table 1: Median rank on the AN-N set of Dinu et al. (2013) (e.g., *personal appeal/charisma*). First row: the A and N are composed and the closest N is returned as a paraphrase. Second row: the N vector is *decomposed* into A and N vectors and their nearest (POS-tag consistent) neighbours are returned.

tors we obtain a median rank that is considerably higher than that of the methods they test.

5 Noun phrase generation

5.1 One-step decomposition

We start with testing one-step decomposition by generating two-word phrases. A first straightforward evaluation consists in decomposing a phrase vector into the correct constituent words. For this purpose, we randomly select (and consequently remove) from the training sets 200 phrases of each type (AN and PN) and apply decomposition operations to 1) their corpus-observed vectors and 2) their composed representations. We generate two words by returning the nearest neighbours (with appropriate POS tags) of the two vectors produced by the decomposition functions. Table 2 reports generation accuracy, i.e., the proportion of times in which we retrieved the correct constituents. The search space consists of the top most frequent 20K nouns, 20K adjectives and 25 prepositions respectively, leading to chance accuracy <0.0001 for nouns and adjectives and <0.05 for prepositions. We obtain relatively high accuracy, with *cbow* vectors consistently outperforming *count* ones. Decomposing composed rather than observed phrase representations is easier, which is to be expected given that composed representations are obtained with a simpler, linear model. Most of the errors consist in generating synonyms (*hard case→difficult case*, *true cost → actual cost*) or related phrases (*stereo speakers→omni-directional sound*).

Next, we use the AN-N dataset of Dinu and colleagues for a more interesting evaluation of one-step decomposition. In particular, we reverse the original paraphrasing direction by attempting to generate, for example, *personal charm* from *charisma*. It is worth stressing the nature of the

Input	Output	**cbow**	**count**
A.N	**A, N**	0.36,0.61	0.20,0.41
P.N	**P, N**	0.93,0.79	0.60,0.57
A∘N	**A, N**	1.00,1.00	0.86,0.99
P∘N	**P, N**	1.00,1.00	1.00,1.00

Table 2: Accuracy of generation models at retrieving (at rank 1) the constituent words of adjective-noun (AN) and preposition-noun (PN) phrases. Observed (**A.N**) and composed representations (**A∘N**) are decomposed with observed- (eq. 2) and composed-trained (eq. 3) functions respectively.

paraphrase-by-generation task we tackle here and in the next experiments. Compositional distributional semantic systems are often evaluated on phrase and sentence paraphrasing data sets (Blacoe and Lapata, 2012; Mitchell and Lapata, 2010; Socher et al., 2011; Turney, 2012). However, these experiments assume a pre-compiled list of candidate paraphrases, and the task is to rank correct paraphrases above foils (paraphrase ranking) or to decide, for a given pair, if the two phrases/sentences are mutual paraphrases (paraphrase detection). Here, instead, we do not assume a given set of candidates: For example, in N→AN paraphrasing, any of $20K^2$ possible combinations of adjectives and nouns from the lexicon could be generated. This is a much more challenging task and it paves the way to more realistic applications of distributional semantics in generation scenarios.

The median ranks of the gold A and N of the Dinu set are shown in the second row of Table 1. As the top-generated noun is almost always, uninterestingly, the input one, we return the next noun. Here we report results for the more motivated corpus-observed training of eq. (2) (unsurprisingly, using composed-phrase training for the task of decomposing *single* nouns leads to lower performance).

Although considerably more difficult than the previous task, the results are still very good, with median ranks under 100 for the *cbow* vectors (random median rank at 10K). Also, the dataset provides only one AN paraphrase for each noun, out of many acceptable ones. Examples of generated phrases are given in Table 3. In addition to generating topically related ANs, we also see nouns disambiguated in different ways than intended in

Input	Output	Gold
reasoning	deductive thinking	abstract thought
jurisdiction	legal authority	legal power
thunderstorm	thundery storm	electrical storm
folk	local music	common people
superstition	old-fashioned religion	superstitious notion
vitriol	political bitterness	sulfuric acid
zoom	fantastic camera	rapid growth
religion	religious religion	religious belief

Table 3: Examples of generating ANs from Ns using the data set of Dinu et al. (2013).

the gold standard (for example *vitriol* and *folk* in Table 3). Other interesting errors consist of decomposing a noun into two words which both have the same meaning as the noun, generating for example *religion → religious religions*. We observe moreover that sometimes the decomposition reflects selectional preference effects, by generating adjectives that denote typical properties of the noun to be paraphrased (e.g., *animosity* is a *(political, personal,...) hostility* or a *fridge* is a *(big, large, small,...) refrigerator*). This effect could be exploited for tasks such as property-based concept description (Kelly et al., 2012).

5.2 Recursive decomposition

We continue by testing generation through *recursive* decomposition on the task of generating noun-preposition-noun (NPN) paraphrases of adjective-nouns (AN) phrases. We introduce a dataset containing 192 AN-NPN pairs (such as *pre-election promises→ promises before election*), which was created by the second author and additionally corrected by an English native speaker. The data set was created by analyzing a list of randomly selected frequent ANs. 49 further ANs (with adjectives such as *amazing* and *great*) were judged not NPN-paraphrasable and were used for the experiment reported in Section 7. The paraphrased subset focuses on preposition diversity and on including prepositions which are rich in semantic content and relevant to paraphrasing the AN. This has led to excluding *of*, which in most cases has the purely syntactic function of connecting the two nouns. The data set contains the following 14 prepositions: *after, against, at, before, between, by, for, from, in, on, per, under, with, without*.[8]

NPN phrase generation involves the application of two decomposition functions. In the first step we decompose using the modifier-noun rule ($f_{\mathrm{decomp_{AN}}}$). We generate a noun from the head slot vector and the "adjective" vector is further decomposed using $f_{\mathrm{decomp_{PN}}}$ (returning the top noun which is not identical to the previously generated one). The results, in terms of top 1 accuracy and median rank, are shown in Table 4. Examples are given in Table 5.

For observed phrase vector training, accuracy and rank are well above chance for all constituents (random accuracy 0.00005 for nouns and 0.04 for prepositions, corresponding median ranks: 10K, 12). Preposition generation is clearly a more difficult task. This is due at least in part to their highly ambiguous and broad semantics, and the way in which they interact with the nouns. For example, *cable through ocean* in Table 5 is a reasonable paraphrase of *undersea cable* despite the gold preposition being *under*. Other than several cases which are acceptable paraphrases but not in the gold standard, phrases related in meaning but not synonymous are the most common error (*overcast skies → skies in sunshine*). We also observe that often the A and N meanings are not fully separated when decomposing and "traces" of the adjective or of the original noun meaning can be found in *both* generated nouns (for example *nearby school → schools after school*). To a lesser degree, this might be desirable as a disambiguation-in-context effect as, for example, in *underground cavern*, *in secret* would not be a context-appropriate paraphrase of *underground*.

6 Noun phrase translation

This section describes preliminary experiments performed in a cross-lingual setting on the task of composing English AN phrases and generating Italian translations.

Creation of cross-lingual vector spaces A common semantic space is required in order to map words and phrases across languages. This problem has been extensively addressed in the bilingual lexicon acquisition literature (Haghighi et al., 2008; Koehn and Knight, 2002). We opt for a very simple yet accurate method (Klementiev et al., 2012; Rapp, 1999) in which a bilingual dictionary is used to identify a set of shared dimensions across spaces and the vectors of both languages are projected into the subspace defined by these (Subspace Projection - SP). This method is applicable to *count*-type vector spaces, for which the dimen-

[8]This dataset is available at `http://clic.cimec.unitn.it/composes`

Input	Output	Training	**cbow**	**count**
A∘N	**N, P, N**	observed	0.98(1),0.08(5.5),0.13(20.5)	0.82(1),0.17(4.5),0.05(71.5)
A∘N	**N, P, N**	composed	0.99(1),0.02(12), 0.12(24)	0.99(1),0.06(10), 0.05(150.5)

Table 4: Top 1 accuracy (median rank) on the AN→NPN paraphrasing data set. AN phrases are composed and then recursively decomposed into N, (P, N). Comma-delimited scores reported for first noun, preposition, second noun in this order. Training is performed on observed (eq. 2) and composed (eq. 3) phrase representations.

Input	Output	Gold
mountainous region	region in highlands	region with mountains
undersea cable	cable through ocean	cable under sea
underground cavern	cavern through rock	cavern under ground
interdisciplinary field	field into research	field between disciplines
inter-war years	years during 1930s	years between wars
post-operative pain	pain through patient	pain after operation
pre-war days	days after wartime	days before war
intergroup differences	differences between intergroup	differences between minorities
superficial level	level between levels	level on surface

Table 5: Examples of generating NPN phrases from composed ANs.

sions correspond to actual words. As the *cbow* dimensions do not correspond to words, we align the *cbow* spaces by using a small dictionary to learn a linear map which transforms the English vectors into Italian ones as done in Mikolov et al. (2013b). This method (Translation Matrix - TM) is applicable to both *cbow* and *count* spaces. We tune the parameters (TM or SP for *count* and dictionary size 5K or 25K for both spaces) on a standard task of translating English words into Italian. We obtain TM-5K for *cbow* and SP-25K for *count* as optimal settings. The two methods perform similarly for low frequency words while *cbow*-TM-5K significantly outperforms *count*-SP-25K for high frequency words. Our results for the *cbow*-TM-5K setting are similar to those reported by Mikolov et al. (2013b).

Cross-lingual decomposition training Training proceeds as in the monolingual case, this time concatenating the training data sets and estimating a single (de)-composition function for the two languages in the shared semantic space. We train both on observed phrase representations (eq. 2) and on composed phrase representations (eq. 3).

Adjective-noun translation dataset We randomly extract 1,000 AN-AN En-It phrase pairs from a phrase table built from parallel movie subtitles, available at `http://opus.lingfil.uu.se/` (OpenSubtitles2012, en-it) (Tiedemann, 2012).

Input	Output	**cbow**	**count**
A∘N(En)	**A,N** (It)	0.31,0.59	0.24,0.54
A∘N (It)	**A,N**(En)	0.50,0.62	0.28,0.48

Table 6: Accuracy of En→It and It→En phrase translation: phrases are composed in source language and decomposed in target language. Training on composed phrase representations (eq. (3)) (with observed phrase training (eq. 2) results are ≈50% lower).

Results are presented in Table 6. While in these preliminary experiments we lack a proper term of comparison, the performance is very good both quantitatively (random < 0.0001) and qualitatively. The En→It examples in Table 7 are representative. In many cases (e.g., *vicious killer*, *rough neighborhood*) we generate translations that are arguably more natural than those in the gold standard. Again, some differences can be explained by different disambiguations (*chest* as *breast*, as in the generated translation, or *box*, as in the gold). Translation into related but not equivalent phrases and generating the same meaning in both constituents (*stellar star*) are again the most significant errors. We also see cases in which this has the desired effect of disambiguating the constituents, such as in the examples in Table 8, showing the nearest neighbours when translating *black tie* and *indissoluble tie*.

Input	Output	Gold
vicious killer	assassino feroce (ferocious killer)	killer pericoloso
spectacular woman	donna affascinante (fascinating woman)	donna eccezionale
huge chest	petto grande (big chest)	scrigno immenso
rough neighborhood	zona malfamata (ill-repute zone)	quartiere difficile
mortal sin	peccato eterno (eternal sin)	pecato mortale
canine star	stella stellare (stellar star)	star canina

Table 7: En→It translation examples (back-translations of generated phrases in parenthesis).

black tie	
cravatta (tie)	**nero** (black)
velluto (velvet)	bianco (white)
giacca (jacket)	giallo (yellow)
indissoluble tie	
alleanza (alliance)	**indissolubile** (indissoluble)
legame (bond)	sacramentale (sacramental)
amicizia (friendship)	inscindibile (inseparable)

Table 8: Top 3 translations of *black tie* and *indissoluble tie*, showing correct disambiguation of *tie*.

	En→It		It→En	
Thr.	Accuracy	Cov.	Accuracy	Cov.
0.00	0.21	100%	0.32	100%
0.55	0.25	70%	0.40	63%
0.60	0.31	32%	0.45	37%
0.65	0.45	9%	0.52	16%

Table 9: AN-AN translation accuracy (*both* A and N correct) when imposing a confidence threshold (random: $1/20K^2$).

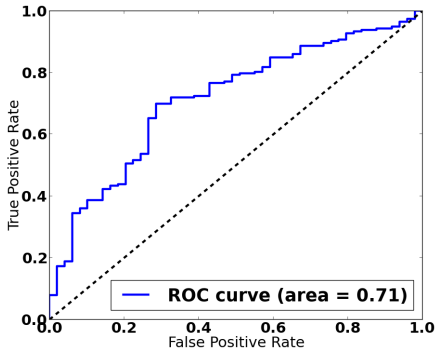


Figure 1: ROC of distinguishing ANs paraphrasable as NPNs from non-paraphrasable ones.

7 Generation confidence and generation quality

In Section 3.2 we have defined a search function s returning a list of lexical nearest neighbours for a constituent vector produced by decomposition. Together with the neighbours, this function can naturally return their similarity score (in our case, the cosine). We call the score associated to the top neighbour the *generation confidence*: if this score is low, the vector has no good match in the lexicon. We observe significant Spearman correlations between the generation confidence of a constituent and its quality (e.g., accuracy, inverse rank) in all the experiments. For example, for the AN(En)→AN(It) experiment, the correlations between the confidence scores and the inverse ranks for As and Ns, for both *cbow* and *count* vectors, range between 0.34 ($p < 1e^{-28}$) and 0.42. In the translation experiments, we can use this to automatically determine a subset on which we can translate with very high accuracy. Table 9 shows AN-AN accuracies and coverage when translating only if confidence is above a certain threshold.

Throughout this paper we have assumed that the syntactic structure of the phrase to be generated is given. In future work we will exploit the correlation between confidence and quality for the purpose of eliminating this assumption. As a concrete example, we can use confidence scores to distinguish the two subsets of the AN-NPN dataset introduced in Section 5: the ANs which are paraphrasable with an NPN from those that do not have this property. We assign an AN to the NPN-paraphrasable class if the mean confidence of the PN expansion in its attempted N(PN) decomposition is above a certain threshold. We plot the ROC curve in Figure 1. We obtain a significant AUC of 0.71.

8 Conclusion

In this paper we have outlined a framework for the task of generation with distributional semantic models. We proposed a simple but effective approach to reverting the composition process to obtain meaningful reformulations of phrases through a synthesis-generation process.

For future work we would like to experiment with more complex models for (de-)composition in order to improve the performance on the tasks we used in this paper. Following this, we

would like to extend the framework to handle arbitrary phrases, including making (confidence-based) choices on the syntactic structure of the phrase to be generated, which we have assumed to be given throughout this paper.

In terms of applications, we believe that the line of research in machine translation that is currently focusing on replacing parallel resources with large amounts of monolingual text provides an interesting setup to test our methods. For example, Klementiev et al. (2012) reconstruct phrase tables based on phrase similarity scores in semantic space. However, they resort to scoring phrase pairs extracted from an aligned parallel corpus, as they do not have a method to freely *generate* these. Similarly, in the recent work on common vector spaces for the representation of images and text, the current emphasis is on retrieving existing captions (Socher et al., 2014) and not actual generation of image descriptions.

From a more theoretical point of view, our work fills an important gap in distributional semantics, making it a bidirectional theory of the connection between language and meaning. We can now translate linguistic strings into vector "thoughts", and the latter into their most appropriate linguistic expression. Several neuroscientific studies suggest that thoughts are represented in the brain by patterns of activation over broad neural areas, and vectors are a natural way to encode such patterns (Haxby et al., 2001; Huth et al., 2012). Some research has already established a connection between neural and distributional semantic vector spaces (Mitchell et al., 2008; Murphy et al., 2012). Generation might be the missing link to powerful computational models that take the neural footprint of a thought as input and produce its linguistic expression.

Acknowledgments

We thank Kevin Knight, Andrew Anderson, Roberto Zamparelli, Angeliki Lazaridou, Nghia The Pham, Germán Kruszewski and Peter Turney for helpful discussions and the anonymous reviewers for their useful comments. We acknowledge the ERC 2011 Starting Independent Research Grant n. 283554 (COMPOSES).

References

Alexandr Andoni and Piotr Indyk. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, January.

Jacob Andreas and Zoubin Ghahramani. 2013. A generative model of vector space semantics. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, pages 91–99, Sofia, Bulgaria.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of EMNLP*, pages 1183–1193, Boston, MA.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of ACL*, To appear, Baltimore, MD.

Marco Baroni. 2013. Composition in distributional semantics. *Language and Linguistics Compass*, 7(10):511–522.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of EMNLP*, pages 546–556, Jeju Island, Korea.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam Khanh Tran. 2012. Distributional semantics in Technicolor. In *Proceedings of ACL*, pages 136–145, Jeju Island, Korea.

Georgiana Dinu, Nghia The Pham, and Marco Baroni. 2013. General estimation and evaluation of compositional distributional semantic models. In *Proceedings of ACL Workshop on Continuous Vector Space Models and their Compositionality*, pages 50–58, Sofia, Bulgaria.

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

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

Emiliano Guevara. 2010. A regression model of adjective-noun compositionality in distributional semantics. In *Proceedings of GEMS*, pages 33–37, Uppsala, Sweden.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proceedings of ACL*, pages 771–779, Columbus, OH, USA, June.

James Haxby, Ida Gobbini, Maura Furey, Alumit Ishai, Jennifer Schouten, and Pietro Pietrini. 2001. Distributed and overlapping representations of faces and objects in ventral temporal cortex. *Science*, 293:2425–2430.

Alexander Huth, Shinji Nishimoto, An Vu, and Jack Gallant. 2012. A continuous semantic space describes the representation of thousands of object and action categories across the human brain. *Neuron*, 76(6):1210–1224.

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

Colin Kelly, Barry Devereux, and Anna Korhonen. 2012. Semi-supervised learning for automatic conceptual property extraction. In *Proceedings of the 3rd Workshop on Cognitive Modeling and Computational Linguistics*, pages 11–20, Montreal, Canada.

Alexandre Klementiev, Ann Irvine, Chris Callison-Burch, and David Yarowsky. 2012. Toward statistical machine translation without parallel corpora. In *Proceedings of EACL*, pages 130–140, Avignon, France.

Philipp Koehn and Kevin Knight. 2002. Learning a translation lexicon from monolingual corpora. In *In Proceedings of ACL Workshop on Unsupervised Lexical Acquisition*, pages 9–16, Philadelphia, PA, USA.

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

Tomas Mikolov, Quoc Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for Machine Translation. `http://arxiv.org/abs/1309.4168`.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL*, pages 236–244, Columbus, OH.

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

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

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

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated english and german corpora. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 519–526. Association for Computational Linguistics.

Richard Socher, Eric Huang, Jeffrey Pennin, Andrew Ng, and Christopher Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of NIPS*, pages 801–809, Granada, Spain.

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

Richard Socher, Quoc Le, Christopher Manning, and Andrew Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*. In press.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey.

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

Peter Turney. 2012. Domain and function: A dual-space model of semantic relations and compositions. *Journal of Artificial Intelligence Research*, 44:533–585.

Fabio Zanzotto, Ioannis Korkontzelos, Francesca Falucchi, and Suresh Manandhar. 2010. Estimating linear models for compositional distributional semantics. In *Proceedings of COLING*, pages 1263–1271, Beijing, China.

Vector space semantics with frequency-driven motifs

Shashank Srivastava
Carnegie Mellon University
Pittsburgh, PA 15217
ssrivastava@cmu.edu

Eduard Hovy
Carnegie Mellon University
Pittsburgh, PA 15217
hovy@cmu.edu

Abstract

Traditional models of distributional semantics suffer from computational issues such as data sparsity for individual lexemes and complexities of modeling semantic composition when dealing with structures larger than single lexical items. In this work, we present a frequency-driven paradigm for robust distributional semantics in terms of semantically cohesive lineal constituents, or *motifs*. The framework subsumes issues such as differential compositional as well as non-compositional behavior of phrasal consituents, and circumvents some problems of data sparsity by design. We design a segmentation model to optimally partition a sentence into lineal constituents, which can be used to define distributional contexts that are less noisy, semantically more interpretable, and linguistically disambiguated. Hellinger PCA embeddings learnt using the framework show competitive results on empirical tasks.

1 Introduction

Meaning in language is a confluence of experientially acquired semantics of words or multi-word phrases, and their semantic composition to create new meanings. For instance, successfully interpreting a sentence such as

The old senator kicked the bucket.

requires the knowledge that the semantic connotations of 'kicking the bucket' as a unit are the same as those for 'dying'. Short of explicit supervision, such semantic mappings must be inferred by a new language speaker through inductive mechanisms operating on observed linguistic usage. This perspective of acquired meaning aligns with the 'meaning is usage' adage, consonant with Wittgenstein's view of semantics. At the same time, the ability to adaptively communicate elaborate meanings can only be conciled through Frege's principle of compositionality, i.e., meanings of larger linguistic constructs can be derived from the meanings of individual components, modulated by their syntactic interrelations. Indeed, most linguistic usage appears compositional. This is supported by the fact even with very limited vocabulary, children and non-native speakers can often communicate surprisingly effectively.

It can be argued that to be sustainable, inductive aspects of meaning must be recurrent enough to be learnable by new users. That is, a non-compositional phrase such as 'kick the bucket' is likely to persist in common parlance only if it is frequently used with its associated semantic mapping. If a usage-driven meaning of a motif is not recurrent enough, learning this mapping is inefficient in two ways. First, the sparseness of observations would severely limit accurate inductive acquisition by new observers. Second, the value of learning a very infrequent semantic mapping is likely marginal. This motivates the need for a frequency-driven view of lexical semantics. In particular, such a perspective can be especially advantageous for distributional semantics for reasons we outline below.

Distributional semantic models (DSMs) that represent words as distributions over neighbouring contexts have been particularly effective in capturing fine-grained lexical semantics (Turney et al., 2010). Such models have engendered improvements in diverse applications such as selectional preference modeling (Erk, 2007), word-sense discrimination (McCarthy and Carroll, 2003), automatic dictionary building (Curran, 2003), and information retrieval (Manning et al., 2008). However, while conventional DSMs consider colloca-

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 634–643,
Baltimore, Maryland, USA, June 23-25 2014.

With the bad press in wake of the financial crisis, businesses are leaving our shores .	
`crisis:`	`<bad, businesses, financial, leaving, press, shores, wake>`
`financial_crisis:`	`<bad press, businesses, in wake of, leaving our shores>`

Table 1: Meaning representation by conventional DSMs vs notional ideal

tion strengths (through counts and PMI scores) of word neighbourhoods, they disregard much of the regularity in human language. Most significantly, word tokens that act as latent dimensions are often derived from arbitrary tokenization. The example given in Table 1 succinctly describes this. The first row in the table shows a representation of the meaning of the token 'crisis' that a conventional DSM might extract from the given sentence after stopword removal. While helpful, the representation seems unsatisfying since words such as 'press', 'wake' and 'shores' seem to have little to do with a crisis. From a semantic perspective, a representation similar to the second is more valuable: not only does it represent a semantic mapping for a more specific meaning, but the latent dimensions of the representation have are less noisy (e.g., while 'wake' is semantically ambiguous, its surrounding context in 'in wake of' disambiguates it) and more intuitive in regards of semantic interepretability. This is the overarching theme of this work: we present a frequency driven paradigm for extending distributional semantics to phrasal and sentential levels in terms of such semantically cohesive, recurrent lexical units or *motifs*.

We propose to identify such semantically cohesive motifs in terms of features inspired from frequency-characteristics, linguistic idiosyncrasies, and shallow syntactic analysis; and explore both supervised and semi-supervised models to optimally segment a sentence into such motifs. Through exploiting regularities in language usage, the framework can efficiently account for both compositional and non-compositional word usage, while avoiding the issue of data-sparsity by design. Our principal contributions in this paper are:

- We present a framework for extending distributional semantics to learn semantic representations of both words and phrases in terms of recurrent motifs, rather than arbitrary word tokens
- We present a simple model to segment a sentence into such motifs using a feature-set drawing from frequency statistics, information theory, linguistic theories and shallow syntactic analysis
- Word and phrasal representations learnt through the approach outperform conventional DSM representations on empirical tasks

This paper is organized as follows: In Section 2, we briefly review related work in the domain of compositional distributional semantics, and motivate our formulation. Section 3 describes our methodology, which consists of a frequency-driven segmentation model to partition text into semantically meaningful recurring lineal-subunits, a representation learning framework for learning new semantic embeddings based on this segmentation, and an approach to use such embeddings in downstream applications. We present experiments and empirical evaluations for our method in Section 4. Finally, we conclude in Section 5 with a summary of our principal findings, and a discussion of possible directions for future work.

2 Related Work

While DSMs have been valuable in representing semantics of single words, approaches to extend them to represent the semantics of phrases and sentences has met with only marginal success. While there is considerable variety in approaches and formulations, existing approaches for phrasal level and sentential semantics can broadly be partitioned into two categories.

2.1 Compositional approaches

These have aimed at using semantic representations for individual words to learn semantic representations for larger linguistic structures. These methods implicitly make an assumption of compositionality, and often include explicit computational models of compositionality. Notable among such models are the additive and multiplicative models of composition by Mitchell and Lapata (2008), Grefenstette et al. (2010), Baroni and

Zamparelli's (2010) model that differentially models content and function words for semantic composition, and Goyal et al.'s SDSM model (2013) that incorporates syntactic roles to model semantic composition. Notable among the most effective distributional representations are the recent deep-learning approaches by Socher et al. (2012), that model vector composition through non-linear transformations. While word embeddings and language models from such methods have been useful for tasks such as relation classification, polarity detection, event coreference and parsing; much of existing literature on composition is based on abstract linguistic theory and conjecture, and there is little evidence to support that learnt representations for larger linguistic units correspond to their semantic meanings. While works such as the SDSM model suffer from the *problem of sparsity* in composing structures beyond bigrams and trigrams, methods such as Mitchell and Lapata (2008)and (Socher et al., 2012) and Grefenstette and Sadrzadeh (2011) are restricted by significant *model biases* in representing semantic composition by generic algebraic operations. Finally, the assumption that semantic meanings for sentences could have representations similar to those for smaller individual tokens is in some sense unintuitive, and not supported by linguistic or semantic theories.

2.2 Tree kernels

Tree Kernel methods have gained popularity in the last decade for capturing syntactic information in the structure of parse trees (Collins and Duffy, 2002; Moschitti, 2006). Instead of procuring explicit representations, the kernel paradigm directly focuses on the larger goal of quantifying semantic similarity of larger linguistic units. Structural kernels for NLP are based on matching substructures within two parse trees , consisting of word-nodes with similar labels. These methods have been useful for eclectic tasks such as parsing, NER, semantic role labeling, and sentiment analysis. Recent approaches such as by Croce et al. (2011) and Srivastava et al. (2013) have attempted to provide formulations to incorporate semantics into tree kernels through the use of distributional word vectors at the individual word-nodes. While this framework is attractive in the lack of assumptions on representation that it makes, the use of distributional embeddings for individual tokens means that it suffers from the same shortcomings as described for the example in Table 1, and hence these methods model semantic relations between word-nodes very weakly. Figure 1 shows an example of the shortcomings of this general approach.

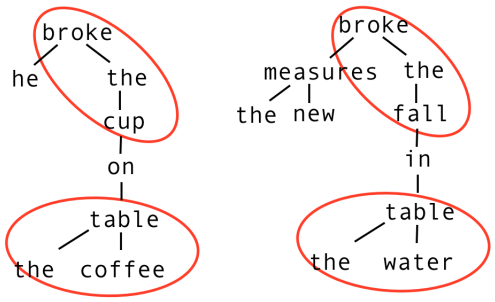


Figure 1: Tokenwise syntactic and semantic similarities don't imply sentential semantic similarity

While the two sentences in consideration have near-identical syntax and could be argued to have semantically aligned words in similar positions, the semantics of the complete sentences are widely divergent. Specifically, the 'bag of words' assumption in tree kernels doesn't suffice for these lexemes, and a stronger semantic model is needed to capture phrasal semantics as well as diverging inter-word relations such as in 'coffee table' and 'water table'. Our hypothesis is that a model that can even weakly identify recurrent motifs such as 'water table' or 'breaking a fall' would be helpful in building more effective semantic representations. A significant advantage of a frequency driven view is that it makes the concern of compositionality of recurrent phrases immaterial. If a motif occurs frequently enough in common parlance, its semantics could be captured with distributional models irrespective of whether its associated semantics are compositional or acquired.

2.3 Identifying multi-word expressions

Several approaches have focused on supervised identification of multi-word expressions (MWEs) through statistical (Pecina, 2008; Villavicencio et al., 2007) and linguistically motivated (Piao et al., 2005) techniques. More recently, hybrid methods based on both statistical as well as linguistic features have been popular (Tsvetkov and Wintner, 2011). Ramisch et al. (2008) demonstrate that adding part-of-speech tags to frequency counts substantially improves performance. Other methods have attempted to exploit morphological, syntactic and semantic characteristics of MWEs. In

particular, approaches such as Bannard (2007) use syntactic rigidity to characterize MWEs. While existing work has focused on the classification task of categorizing a phrasal constituent as a MWE or a non-MWE, the general ideas of most of these works are in line with our current framework, and the feature-set for our motif segmentation model is designed to subsume most of these ideas. It is worthwhile to point out that the task of motif segmentation is slightly different from MWE identification. Specifically, the onus on recurrent occurrences means that non-decomposibility is not an essential consideration for a word to be considered a motif. In line with the proposed paradigm, typical MWEs such as 'shoot the breeze', 'sour note' and 'hot dog' would be considered valid lineal motifs. [1] In addition, even decomposable recurrent lineal phrases such as 'love story', 'federal government', and 'millions of people' are marked as meaningful recurrent motifs. Finally, and least interestingly, we include common named entities such as 'United States' and 'Java Virtual Machine' within the ambit of motifs.

3 Method

In this section, we define our frequency-driven framework for distributional semantics in detail. As just described above, our definition for motifs is less specific than MWEs. With such a working definition, contiguous motifs are likely to make distributional representations less noisy and also assist in disambiguating context. Also, the lack of specificity ensures that such motifs are common enough to meaningfully influence distributional representation beyond single tokens. A method towards frequency-driven distributional semantics could involve the following principal components:

3.1 Linear segmentation model

The segmentation model forms the core of the framework. Ideally, it fragments a given sentence into non-overlapping, semantically meaningful, empirically frequent contiguous sub-units or motifs. The model accounts for possible segmentations of a sentence into potential motifs, and prefers recurrent and cohesive motifs through features that capture frequency-based and statistical features, as well as linguistic idiosyncracies. This is accomplished using a very simple linear chain model and a rich feature set consisting of a combination of frequency-driven, information theoretic and linguistically motivated features.

Let an observed sentence be denoted by $\mathbf{x}$, with the individual tokens x_i denoting the i'th token in the sentence. The segmentation model is a chain LVM (latent variable model) that aims to maximize a linear objective defined by:

$$J = \sum_i w_i f_i(y_k, y_{k-1}, \mathbf{x})$$

where f_i are arbitrary Markov features that can depend on segments (potential motifs) of the observed sentence $\mathbf{x}$, and contiguous latent states. The features are chosen so as to best represent frequency-based, statistical as well as linguistic considerations for treating a segment as an agglutinative unit, or a motif. In specific, these features could encode characteristics such as frequency statistics, collocation strengths and syntactic distinctness, or inflectional rigidity of the considered segments; described in detail in Section 3.2. The model is an instantiation of a simple featurized HMM, and the weighted sum of features corresponding to a segment is cognate with an affinity score for the 'stickiness' of the segment, i.e., the affinity for the segment to be treated as holistic unit or a single motif.

We also associate a penalizing cost for each non unary-motif to avoid aggressive agglutination of tokens. In particular, for an ngram occurrence to be considered a motif, the marginal contribution due to the affinity of the prospective motif should at minimum exceed this penalty. The weights for the affinity functions as well as these penalties are learnt from data using full as well as partial annotations. The latent state-variables y_k denotes the membership of the token $\mathbf{x_k}$ to a unary or a larger motif; and the state-sequence collectively gives the segmentation of the sentence. An individual state-variable y_k encodes a pairing of the size of the encompassing ngram motif, and the position of the word x_k within it. For instance, $y_k = T_3$ denotes that the token $\mathbf{x_k}$ is the final position in a trigram motif.

3.1.1 Inference of optimal segmentation

If the optimal weights w_i are known, inference for the best motif segmentation can be performed

[1] We note that since we take motifs as lineal units, the current method doesn't subsume several common non-contiguous MWEs such as 'let off' in 'let him off'.

in linear time (in the number of tokens) following the generalized Viterbi algorithm. A slightly modified version of Viterbi could also be used to find segmentations that are constrained to agree with some given motif boundaries, but can segment other parts of the sentence optimally under these constraints. This is necessary for the scenario of semi-supervised learning of weights with partially annotated sentences, as described later.

3.2 Learning motif affinities and penalties

We briefly discuss data-driven learning of weights for features that define the motif affinity scores and penalties. We describe learning of the model parameters with fully annotated training data, as well as an approach for learning motif segmentation that requires only partial supervision.

Supervised learning: In the supervised case, optimal state sequences $\mathbf{y^{(k)}}$ are fully observed for the training set. For this purpose, we created a dataset of 1000 sentences from the Simple English Wikipedia and the Gigaword Corpus, and manually annotated it with motif boundaries using BRAT (Stenetorp et al., 2012). In this case, learning can follow the online structured perceptron learning procedure by Collins (2002), where weights updates for the k'th training example $(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})$ are given as:

$$w_i \leftarrow w_i + \alpha(f_i(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}) - f_i(\mathbf{x}^{(k)}, \mathbf{y}'))$$

Here $\mathbf{y}' = Decode(\mathbf{x}^{(k)}, \mathbf{w})$ is the optimal Viterbi decoding using the current estimates of the weights. Updates are run for a large number of iterations until the change in objective drops below a threshold, and the learning rate α is adaptively modified as described in Collins et al. Implicitly, the weight learning algorithm can be seen as a gradient descent procedure minimizing the difference between the scores of highest scoring (Viterbi) state sequences, and the label state sequences.

Semi-supervised learning: In the semi-supervised case, the labels $y_i^{(k)}$ are known only for some of the tokens in $\mathbf{x^{(k)}}$. This is a commonplace scenario, where a part of a sentence has clear motif-boundaries, whereas the rest of the sentence is not annotated. For accumulating such data, we looked for occurrences of 2500 expressions from the WikiMWE dataset in sentences from the combined Simple English Wikipedia and Gigaword corpora. The query expressions in the retrieved sentences were marked with motif boundaries, while the remaining tokens in the sentences were left unannotated.

While the Viterbi algorithm can be used for tagging optimal state-sequences given the weights, the structured perceptron can learn optimal model weights given gold-standard sequence labels. Hence, in this case, we use a variation of the hard EM algorithm for learning. The algorithm proceeds as follows: in the E-step, we use the current values of weights to compute hard-expectations, i.e., the best scoring Viterbi sequences among those consistent with the observed state labels. In the M-step, we take the decoded state-sequences in the E-step as observed, and run perceptron learning to update feature weights w_i. Pseudocode of the learning algorithm for the partially labeled case is given in Algorithm 1.

Algorithm 1

1: **Input:** Partially labeled data $D = \{(x, y)_i\}$
2: **Output:** Weights w
3: **Initialization:** Set w_i randomly, $\forall i$
4: **for** $i : 1$ to $maxIter$ **do**
5: Decode D with current w to find optimal Viterbi paths that agree with (partial) ground truths.
6: Run Structured Perceptron algorithm with decoded tag-sequences to update weights w
7: **end for**
8: return w

The semi-supervised approach enables incorporation of significantly more training data. In particular, this method could be used in conjunction with a supervised approach. This would involve initializing the weights prior to the semi-supervised procedure with the weights from the supervised learning model, so as to seed the semi-supervised approach with reasonable model, and use the partially annotated data to fine-tune the supervised model. The sequential approach, akin to annealing weights, can efficiently utilize both full and partial annotations.

3.2.1 Feature engineering

In this section, we describe the principal features used in the segmentation model
Transitional features and penalties:

- Transitional features $f_{trans}(y_{i-1}, y_i) =$

I_{y_{i-1},y_i} [2] describing the transitional affinities of state pairs. Since our state definitions preclude certain transitions (such as from state T_2 to T_1), these weights are initialized to $-\infty$ to expedite training.

- N-gram penalties: f_{ngram} We define a penalty for tagging each non-unary motif as described before. For a motif to be tagged, the improvement in objective score should at least exceed the corresponding penalty. e.g., $f_{qgram}(y_i) = I_{y_i=Q_4}$ denotes the penalty for tagging a tetragram. [3]

Frequency-based, information theoretic, and POS features:

- Absolute and log-normalized motif frequencies $f_{ngram}(x_{i-n+1}, ...x_{i-1}, x_i, y_i)$. This feature is associated with a particular token-sequence and ngram-tag, and takes the value of the motif-frequency if the motif token-sequence matches the feature token-sequence, and is marked as with a matching tag. e.g., $f_{bgram}(x_{i-1} = love, x_i = \text{story}, y_i = B_2)$.

- Absolute and log-normalized motif frequencies for a particular POS-sequence. This feature is associated with a particular POS-tag sequence and ngram-tag, and takes the value of the motif-frequency if the motif token-sequence gets a matching tag, and is marked as with a matching ngram tag. e.g., $f_{bgram}(p_{i-1} = VB, p_i = NN, y_i = B_2)$.

- Medians and maxima of pairwise collocation statistics for tokens for a particular size of ngram motifs: we use the following statistics: pointwise mutual information, Chi-square statistic, and conditional probability. We also used POS sensitive versions of these, which performed much better than plain versions in our evaluations.

- Histogram counts of inflectional forms of token sequence for the corresponding ngram motif and POS sequence: this features takes the value of the count of inflectional forms of an ngram that account for 90% of occurrences of all inflectional forms.

- Entropies of histogram distributions of inflectional variants (described above).

- Features encoding syntactic rigidity: ratios and log-ratios of frequencies of an ngram motif and variations by replacing a token using near synonyms from its synset.

Additionally, a few feature for the segmentations model contained minor orthographic features based on word shape (length and capitalization patterns). Also, all numbers, URLs, and currency symbols were normalized to the special NUMERIC, URL, and CURRENCY tokens respectively. Finally, a gazetteer feature checked for occurrences of motifs in a gazetteer of named entities.

3.3 Representation learning

With the segmentation model described in the previous section, we process text from the English Gigaword corpus and the Simple English Wikipedia to partition sentences into motifs. Since the segmentation model accounts for the contexts of the entire sentence in determining motifs, different instances of the same token could evoke different meaning representations. Consider the following sentences tagged by the segmentation model, that would correspond to different representations of the token 'remains': once as a standalone motif, and once as part of an encompassing bigram motif ('remains classified').

Hog prices have declined sharply , while the cost of corn remains relatively high.

Even with the release of such documents, questions are not answered, since only the agency knows what remains classified

Given constituent motifs of each sentence in the data, we can now define neighbourhood distributions for unary or phrasal motifs in terms of other motifs (as envisioned in Table 1). In our experiments, we use a window-length of 5 adjoining motifs on either side to define the neighbourhood of a constituent. Naturally, in the presence of multi-word motifs, the neighbourhood boundary could be more extended than in a conventional DSM.

With such neighbourhood contexts, the distributional paradigm posits that semantic similarity between a pair of motifs can be given by a sense of 'distance' between the two distributions. Most popularly, traditional measures of vector distance

[2] Here, I denotes the indicator function

[3] It is straightforward to preclude partial n-gram annotations near sentence boundaries with prohibitive penalties.

such as the cosine similarity, Euclidean distance and City-block distance have been used in several distributional approaches. Additionally, several distance measures between discrete distributions exist in statistical literature, most famously the Kullback Leibler divergence, Bhattacharyya distance and the Hellinger distance. Recent work (Lebret and Lebret, 2013) has shown that the Hellinger distance is an especially effective measure in learning distributional embeddings, with Hellinger PCA being much more computationally inexpensive than neural language modeling approaches, while performing much better than standard PCA, and competitive with the state-of-the-art in downstream evaluations. Hence, we use the Hellinger measure between neighbourhood motif distributions in learning representations.

The Hellinger distance between two categorical distributions $P = (p_1...p_k)$ and $Q = (q_1...q_k)$ is defined as:

$$H(P,Q) = \frac{1}{\sqrt{2}}\sqrt{\sum_{i=1}^{k}(\sqrt{p_i} - \sqrt{q_i})^2}$$
$$= \frac{1}{\sqrt{2}}\left\|\sqrt{P} - \sqrt{Q}\right\|_2$$

The Hellinger measure has intuitively desirable properties: specifically, it can be seen as the Euclidean distance between the square-roots transformed distributions, where both vectors $\sqrt{P}$ and $\sqrt{Q}$ are length-normalized under the same(Euclidean) norm. Finally, we perform SVD on the motif similarity matrix (with size of the order of the total vocabulary in the corpus), and retain the first k principal eigenvectors to obtain low-dimensional vector representations that are more convenient to work with. In our preliminary experiments, we found that $k = 300$ gave quantitatively good results, with marginal change with added dimensionality. We use this setting for all our experiments.

4 Experiments

In this section, we describe some experimental evaluations and findings for our approach. We first quantitatively and qualitatively analyze the performance of the segmentation model, and then evaluate the distributional motif representations learnt by the model through two downstream applications.

4.1 Motif segmentation

In an evaluation of the motif segmentations model within the perspective of our framework, we believe that exact correspondence to human judgment is unrealistic, since guiding principles for defining motifs, such as semantic cohesion, are hard to define and only serve as working principles. However, for purposes of relative comparison, we quantitatively evaluate the performance of the motif segmentation models on the fully annotated dataset. For this experiment, the gold-annotated corpus was split into a training and test sets in a 9:1 proportion. A small fraction of the training split was set apart for development and validation. For this evaluation, we considered a motif boundary as correct only for an exact match, i.e., when both its boundaries (left and right) were correctly predicted. Also, since a majority of motifs are unary tokens, including them into consideration artificially boosts the accuracy, whereas we are more interested in the prediction of larger n-gram tokens. Hence we report results on the performance on only non-unary motifs.

	P	**R**	**F**
Rule-based baseline	0.85	0.10	0.18
Supervised	0.62	0.28	0.39
Semi-supervised	0.30	0.17	0.22
Supervised + annealing	0.69	0.38	0.49

Table 2: Results for motif segmentations

Table 2 shows the performance of the segmentation model with the three proposed learning approaches described earlier. For a baseline, we consider a rule-based model that simply learns all ngram segmentations seen in the training data, and marks any occurrence of a matching token sequence as a motif; without taking neighbouring context into account. We observe that this model has a very high precision (since many token sequences marked as motifs would recur in similar contexts, and would thus have the same motif boundaries). However, the rule-based method has a very row recall due to lack of generalization capabilities. We see that while all three learning algorithms perform better than the baseline, the performance of the purely unsupervised system is inferior to supervised approaches. This is not unexpected: the supervision provided to the model is very weak due to a lack of negative examples (which leads to spurious motif taggings,

While men often (openly or privately) sympathized with Prince Charles when the princess went public about her rotten marriage , women cheered her on.
The healthcare initiative has become a White elephant for the federal government.
Chirac and Juppe have made a bad situation worse by seeking to meet Maastricht criteria not by cutting spending, but by raising taxes still further.
Now , say Vatican observers , Pope John Paul II wants to show the world that many church members did resist the Third Reich and paid the price.

Table 3: Examples of output from sentence segmentation model

leading to a low precision), as well as no examples of transitions between adjacent motifs (to learn transitional weights and penalties). The supervised model expectedly outperforms both the rule-based and the semi-supervised systems. However, the supervised learning model with subsequent annealing outperforms the supervised model in terms of both precision and recall; showing the utility of the semi-supervised method when seeded with a good initial model, and the additive value of partially labeled data.

Qualitative analysis of motif-segmented sentences shows that our designed feature-set is effective and helpful in identifying semantically cohesive ngrams. Table 3 provides four examples. The first example correctly identifies 'went public', while missing out on the potential motif 'cheered her on'. In general, these examples illustrate that the model can identify idiomatic and idiosyncratic themes as well as commonly recurrent ngrams (in the second example, the model picks out 'has become' which is highly recurrent, but doesn't have the semantic cohesiveness of some of the other motifs). In particular, consider the second example, where the model picks 'white elephant' as a motif. In such cases, the disambiguating influence of context incorporated by the motif is apparent.

Elephant	White elephant
tusks	expensive
trunk	spend
african	biggest
white	the project
indian	very high
baby	multibillion dollar

The above table shows some of the top results for the unary token 'elephant' by frequency, and frequent unary and non-unary motifs for the motif 'white elephant' retrieved by the segmentation model.

4.2 Distributional representations

For evaluating distributional representations for motifs (in terms of other motifs) learnt by the framework, we test these representations in two downstream tasks: sentence polarity classification and metaphor detection. For sentence polarity, we consider the Cornell Sentence Polarity corpus by Pang and Lee (2005), where the task is to classify the polarity of a sentence as positive or negative. The data consists of 10662 sentences from movie reviews that have been annotated as either positive or negative. For composing the motifs representations to get judgments on semantic similarity of sentences, we use our recent Vector Tree Kernel approach The VTK approach defines a convolutional kernel over graphs defined by the dependency parses of sentences, using a vector representation at each graph node that representing a single lexical token. For our purposes, we modify the approach to merge the nodes of all tokens that constitute a motif occurrence, and use the motif representation as the vector associated with the node. Table 4 shows results for the sentence polarity task.

	P	**R**	**F1**
DSM	0.56	0.50	0.53
AVM	0.55	0.53	0.54
MVM	0.55	0.49	0.52
VTK	0.65	0.58	0.62
VTK + MotifDSM	**0.66**	**0.60**	**0.63**

Table 4: Results for Sentence Polarity detection

For this task, the motif based distributional embeddings vastly outperform a conventional distributional model (DSM) based on token distributions, as well as additive (AVM) and multiplicative (MVM) models of vector compositionality, as

proposed by Lapata et al. The model is competitive with the state-of-the-art VTK (Srivastava et al., 2013) that uses the SENNA neural embeddings by Collobert et al. (2011).

	P	R	F1
CRF	**0.74**	0.50	0.59
SVM+DSM	0.63	0.80	0.71
VTK+ SENNA	0.67	**0.87**	0.76
VTK+ MotifDSM	0.70	**0.87**	**0.78**

Table 5: Results for Metaphor identification

On the metaphor detection task, we use the Metaphor dataset (Hovy et al., 2013). The data consists of sentences with defined phrases, and the task consists of identifying the linguistic use in these phrases as metaphorical or literal. For this task, the motif based model is expected to perform well as common metaphorical usage is generally through idiosyncratic MWEs, which the motif based models is specially geared to capture through the features of the segmentation model. For this task, we again use the VTK formalism for combining vector representations of the individual motifs. Table 5 shows that the motif-based DSM does better than discriminative models such as CRFs and SVMs, and also slightly improves on the VTK kernel with distributional embeddings.

5 Conclusion

We have presented a new frequency-driven framework for distributional semantics of not only lexical items but also longer cohesive motifs. The theme of this work is a general paradigm of seeking motifs that are recurrent in common parlance, are semantically coherent, and are possibly non-compositional. Such a framework for distributional models avoids the issue of data sparsity in learning of representations for larger linguistic structures. The approach depends on drawing features from frequency statistics, statistical correlations, and linguistic theories; and this work provides a computational framework to jointly model recurrence and semantic cohesiveness of motifs through compositional penalties and affinity scores in a data driven way.

While being deliberately vague in our working definition of motifs, we have presented simple efficient formulations to extract such motifs that uses both annotated as well as partially unannotated data. The qualitative and quantitative analyis of results from our preliminary motif segmentation model indicate that such motifs can help to disambiguate contexts of single tokens, and provide cleaner, more interpretable representations. Finally, we obtain motif representations in form of low-dimensional vector-space embeddings, and our experimental findings indicate value of the learnt representations in downstream applications. We believe that the approach has considerable theoretical as well as practical merits, and provides a simple and clean formulation for modeling phrasal and sentential semantics.

In particular, we believe that ours is the first method that can invoke different meaning representations for a token depending on textual context of the sentence. The flexibility of having separate representations to model different semantic senses has considerable valuable, as compared with extant approaches that assign a single representation to each token, and are hence constrained to conflate several semantic senses into a common representation. The approach also elegantly deals with the problematic issue of differential compositional and non-compositional usage of words. Future work can focus on a more thorough quantitative evaluation of the paradigm, as well as extension to model non-contiguous motifs.

References

Colin Bannard. 2007. A measure of syntactic flexibility for automatically identifying multiword expressions in corpora. In *Proceedings of the Workshop on a Broader Perspective on Multiword Expressions*, pages 1–8. Association for Computational Linguistics.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193. Association for Computational Linguistics.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 263–270. Association for Computational Linguistics.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics.

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

Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2011. Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1034–1046. Association for Computational Linguistics.

James Richard Curran. 2003. *From Distributional to Semantic Similarity*. Ph.D. thesis, Institute for Communicating and Collaborative Systems School of Informatics University of Edinburgh.

Katrin Erk. 2007. A simple, similarity-based model for selectional preferences. In *ACL*.

Kartik Goyal, Sujay Kumar Jauhar, Huiying Li, Mrinmaya Sachan, Shashank Srivastava, and Eduard Hovy. 2013. A structured distributional semantic model: Integrating structure with semantics. *ACL 2013*, page 20.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1394–1404. Association for Computational Linguistics.

Edward Grefenstette, Mehrnoosh Sadrzadeh, Stephen Clark, Bob Coecke, and Stephen Pulman. 2010. Concrete sentence spaces for compositional distributional models of meaning. *arXiv preprint arXiv:1101.0309*.

Dirk Hovy, Shashank Srivastava, Sujay Kumar Jauhar, Mrinmaya Sachan, Kartik Goyal, Huiying Li, Whitney Sanders, and Eduard Hovy. 2013. Identifying metaphorical word use with tree kernels. *Meta4NLP 2013*, page 52.

Rémi Lebret and Ronan Lebret. 2013. Word emdeddings through hellinger pca. *arXiv preprint arXiv:1312.5542*.

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge.

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

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *ACL*, pages 236–244.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *Machine Learning: ECML 2006*, pages 318–329. Springer.

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

Pavel Pecina. 2008. A machine learning approach to multiword expression extraction. In *Proceedings of the LREC MWE 2008 Workshop*, pages 54–57. Citeseer.

Scott Songlin Piao, Paul Rayson, Dawn Archer, and Tony McEnery. 2005. Comparing and combining a semantic tagger and a statistical tool for mwe extraction. *Computer Speech & Language*, 19(4):378–397.

Carlos Ramisch, Paulo Schreiner, Marco Idiart, and Aline Villavicencio. 2008. An evaluation of methods for the extraction of multiword expressions. In *Proceedings of the LREC Workshop-Towards a Shared Task for Multiword Expressions (MWE 2008)*, pages 50–53.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics.

Shashank Srivastava, Dirk Hovy, and Eduard H. Hovy. 2013. A walk-based semantically enriched tree kernel over distributed word representations. In *EMNLP*, pages 1411–1416.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. Brat: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107. Association for Computational Linguistics.

Yulia Tsvetkov and Shuly Wintner. 2011. Identification of multi-word expressions by combining multiple linguistic information sources. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 836–845. Association for Computational Linguistics.

Peter D Turney, Patrick Pantel, et al. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37(1):141–188.

Aline Villavicencio, Valia Kordoni, Yi Zhang, Marco Idiart, and Carlos Ramisch. 2007. Validation and evaluation of automatically acquired multiword expressions for grammar engineering. In *EMNLP-CoNLL*, pages 1034–1043.

Lexical Inference over Multi-Word Predicates: A Distributional Approach

Omri Abend Shay B. Cohen Mark Steedman

School of Informatics, University of Edinburgh,
Edinburgh EH8 9AB, United Kingdom
{oabend,scohen,steedman}@inf.ed.ac.uk

Abstract

Representing predicates in terms of their argument distribution is common practice in NLP. Multi-word predicates (MWPs) in this context are often either disregarded or considered as fixed expressions. The latter treatment is unsatisfactory in two ways: (1) identifying MWPs is notoriously difficult, (2) MWPs show varying degrees of compositionality and could benefit from taking into account the identity of their component parts. We propose a novel approach that integrates the distributional representation of multiple sub-sets of the MWP's words. We assume a latent distribution over sub-sets of the MWP, and estimate it relative to a downstream prediction task. Focusing on the supervised identification of lexical inference relations, we compare against state-of-the-art baselines that consider a single sub-set of an MWP, obtaining substantial improvements. To our knowledge, this is the first work to address lexical relations between MWPs of varying degrees of compositionality within distributional semantics.

1 Introduction

Multi-word expressions (MWEs) constitute a large part of the lexicon and account for much of its growth (Jackendoff, 2002; Seaton and Macaulay, 2002). However, despite their importance, MWEs remain difficult to define and model, and consequently pose serious difficulties for NLP applications (Sag et al., 2001). Multi-word Predicates (MWPs; sometimes termed Complex Predicates) form an important and much addressed subclass of MWEs and are the focus of this paper.

MWPs are informally defined as multiple words that constitute a single predicate (Alsina et al., 1997). MWPs encompass a wide range of phenomena, including causatives, light verbs, phrasal verbs, serial verb constructions and many others, and pose considerable challenges to both linguistic theory and NLP applications (see Section 2). Part of the difficulty in treating them stems from their position on the borderline between syntax and the lexicon. It is therefore often unclear whether they should be treated as fixed expressions, as compositional phrases that reflect the properties of their component parts or as both.

This work addresses the modelling of MWPs within the context of distributional semantics (Turney and Pantel, 2010), in which predicates are represented through the distribution of arguments they may take. In order to collect meaningful statistics, the predicate's lexical unit should be sufficiently frequent and semantically unambiguous.

MWPs pose a challenge to such models, as naïvely collecting statistics over all instances of highly ambiguous verbs is likely to result in noisy representations. For instance, the verb "take" may appear in MWPs as varied as "take time", "take effect" and "take to the hills". This heterogeneity of "take" is likely to have a negative effect on downstream systems that use its distributional representation. For instance, while "take" and "accept" are often considered lexically similar, the high frequency in which "take" participates in non-compositional MWPs is likely to push the two verbs' distributional representations apart.

A straightforward approach to this problem is to represent the predicate as a conjunction of multiple words, thereby trading ambiguity for sparsity. For instance, the verb "take" could be conjoined with its object (e.g., "take care", "take a bus"). This approach, however, raises the challenge of identifying the sub-set of the predicate's words that should be taken to represent it (henceforth, its *lexical components* or LCs).

We propose a novel approach that addresses this

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 644–654,
Baltimore, Maryland, USA, June 23-25 2014.

challenge in the context of identifying lexical inference relations between predicates (Lin and Pantel, 2001; Schoenmackers et al., 2010; Melamud et al., 2013a, *inter alia*). A (lexical) inference relation $p_L \rightarrow p_R$ is said to hold if the relation denoted by p_R generally holds between a set of arguments whenever the relation p_L does. For instance, an inference relation holds between "annex" and "control" since if a country annexes another, it generally controls it. Most works to this task use distributional similarity, either as their main component (Szpektor and Dagan, 2008; Melamud et al., 2013b), or as part of a more comprehensive system (Berant et al., 2011; Lewis and Steedman, 2013).

For example, consider the verb "take". While the inference relation "*have* → *take*" does not generally hold, it does hold in the case of some light verbs, such as "*have a look* → *take a look*", underscoring the importance of taking more inclusive LCs into account. On the other hand, the predicate "likely to give a green light" is unlikely to appear often even within a very large corpus, and could benefit from taking its lexical sub-units (e.g., "likely" or "give a green light") into account.

We present a novel approach to the task that models the selection and relative weighting of the predicate's LCs using latent variables. This approach allows the classifier that uses the distributional representations to take into account the most relevant LCs in order to make the prediction. By doing so, we avoid the notoriously difficult problem of defining and identifying MWPs and account for predicates of various sizes and degrees of compositionality. To our knowledge, this is the first work to address lexical relations between MWPs of varying degrees of compositionality within distributional semantics.

We conduct experiments on the dataset of Zeichner et al. (2012) and compare our methods with analogous ones that select a fixed LC, using state-of-the-art feature sets. Our method obtains substantial performance gains across all scenarios.

Finally, we note that our approach is cognitively appealing. Significant cognitive findings support the claim that a speaker's lexicon consists of partially overlapping lexical units of various sizes, of which several can be evoked in the interpretation of an utterance (Jackendoff, 2002; Wray, 2008).

2 Background and Related Work

Inference Relations. The detection of inference relations between predicates has become a central task over the past few years (Sekine, 2005; Zanzotto et al., 2006; Schoenmackers et al., 2010; Berant et al., 2011; Melamud et al., 2013a, *inter alia*). Inference rules are used in a wide variety of applications including Question Answering (Ravichandran and Hovy, 2002), Information Extraction (Shinyama and Sekine, 2006), and as a main component in Textual Entailment systems (Dinu and Wang, 2009; Dagan et al., 2013).

Most approaches to the task used distributional similarity as a major component within their system. Lin and Pantel (2001) introduced DIRT, an unsupervised distributional system for detecting inference relations. The system is still considered a state-of-the-art baseline (Melamud et al., 2013a), and is often used as a component within larger systems. Schoenmackers et al. (2010) presented an unsupervised system for learning inference rules directly from open-domain web data. Melamud et al. (2013a) used topic models to combine type-level predicate inference rules with token-level information from their arguments in a specific context. Melamud et al. (2013b) used lexical expansion to improve the representation of infrequent predicates. Lewis and Steedman (2013) combined distributional and symbolic representations, evaluating on a Question Answering task, as well as on a quantification-focused entailment dataset.

Several studies tackled the task using supervised systems. Weisman et al. (2012) used a set of linguistically motivated features, but evaluated their system on a corpus that consists almost entirely of single-word predicates. Mirkin et al. (2006) presented a system for learning inference rules between nouns, using distributional similarity and pattern-based features. Hagiwara et al. (2009) identified synonyms using a supervised approach relying on distributional and syntactic features. Berant et al. (2011) used distributional similarity between predicates to weight the edges of an entailment graph. By imposing global constraints on the structure of the graph, they obtained a more accurate set of inference rules.

Previous work used simple methods to select the predicate's LC. Some filtered out frequent highly ambiguous verbs (Lewis and Steedman, 2013), others selected a single representative word (Melamud et al., 2013a), while yet others used multi-word LCs but treated them as fixed expressions (Lin and Pantel, 2001; Berant et al., 2011).

The goals of the above studies are largely com-

plementary to ours. While previous work focused either on improving the quality of the distributional representations themselves or on their incorporation into more elaborate systems, we focus on the integration of the distributional representation of multiple LCs to improve the identification of inference relations between MWPs.

MWP Extraction and Identification. MWPs have received considerable attention over the years in both theoretical and applicative contexts. Their position on the crossroads of syntax and the lexicon, their varying degrees of compositionality, as well as the wealth of linguistic phenomena they exhibit, made them the object of ongoing linguistic discussion (Alsina et al., 1997; Butt, 2010).

In NLP, the discovery and identification of MWEs in general and MWPs in particular has been the focus of much work over the years (Lin, 1999; Baldwin et al., 2003; Biemann and Giesbrecht, 2011). Despite wide interest, the field has yet to converge to a general and widely agreed-upon method for identifying MWPs. See (Ramisch et al., 2013) for an overview.

Most work on MWEs emphasized idiosyncratic or non-compositional expressions. Other lines of work focused on specific MWP classes such as light verbs (Tu and Roth, 2011; Vincze et al., 2013) and phrasal verbs (McCarthy et al., 2003; Pichotta and DeNero, 2013). Our work proposes a uniform treatment to MWPs of varying degrees of compositionality, and avoids defining MWPs explicitly by modelling their LCs as latent variables.

Compositional Distributional Semantics. Much work in recent years has concentrated on the relation between the distributional representations of composite phrases and the representations of their component sub-parts (Widdows, 2008; Mitchell and Lapata, 2010; Baroni and Zamparelli, 2010; Coecke et al., 2010). Several works have used compositional distributional semantics (CDS) representations to assess the compositionality of MWEs, such as noun compounds (Reddy et al., 2011) or verb-noun combinations (Kiela and Clark, 2013). Despite significant advances, previous work has mostly been concerned with highly compositional cases and does not address the distributional representation of predicates of varying degrees of compositionality.

3 Our Proposal: A Latent LC Approach

This section details our approach for distributionally representing MWPs by leveraging their component LCs. Section 3.1 describes our general approach, Section 3.2 presents our model and Section 3.3 details the feature set.

3.1 General Approach and Notation

We propose a method for addressing MWPs of varying degrees of compositionality through the integration of the distributional representation of multiple sub-sets of the predicate's words (LCs). We use it to tackle a supervised prediction task that represents predicates distributionally. Our model assumes a latent distribution over the LCs, and estimates its parameters so to best conform to the goals of the target prediction task.

Formally, given a predicate p, we denote the set of words comprising it as $W(p)$. The set of allowable LCs for p is denoted with $H_p \subset 2^{W(p)}$. H_p contains all sub-sets of p that we consider as apriori possible to represent p. For instance, if p is "likely to give a green light", H_p may include LCs such as "likely" or "give light". As our method is aimed at discovering the most relevant LCs, we do not attempt to analyze the MWPs in advance, but rather take an inclusive H_p, allowing the model to estimate the relative weights of the LCs.

The task we use as a testbed for our approach is the lexical inference identification task between predicates. Given a pair of predicates $p = (p_L, p_R)$, the task is to predict whether an inference relation holds between them. For instance, if p_L is "devour" and p_R is "eat greedily", the classifier should use the similarity between "devour" and "eat" in order to correctly predict an inference relation in this case. Selecting the wider LC "eat greedily" might result in sparser statistics. In other examples, however, taking a wider LC is potentially beneficial. For instance, the dissimilarity between "take" and "make" should not prevent the classifier from identifying the inference relation between "take a step" and "make a step".

Our statistical model aims at predicting the correct label by making use of partially overlapping LCs of various sizes, both for the premise left-hand side (LHS) predicate p_L and the hypothesis right-hand side (RHS) predicate p_R. More formally, we take the space of values for our latent LC variables to be $H_{p_L,p_R} = H_{p_L} \times H_{p_R}$.

Our evaluation dataset consists of pairs $p^{(i)} = (p_L^{(i)}, p_R^{(i)})$ for $i \in \{1, \ldots, M\}$, where M is the number of examples available, coupled with their gold-standard labels $y^{(i)} \in \{1, -1\}$. For brevity, we denote $H^{(i)} = H_{p^{(i)}} = H_{p_L^{(i)}, p_R^{(i)}}$. We also as-

sume the existence of a feature function $\Phi(p, y, h)$ which maps a triplet of a predicate pair p, an inference label y, and a latent state $h \in H_p$ to $\mathbb{R}^d$ for some integer d. We denote the training set by $\mathcal{D}$.

3.2 The Model

We address the task with a latent variable log-linear model, representing the LCs of the predicates. We choose this model for its generality, conceptual simplicity, and because it allows to easily incorporate various feature sets and sets of latent variables. We introduce L_2 regularization to avoid over-fitting. We use maximum likelihood estimation, and arrive at the following objective function:

$$L(w|\mathcal{D}) = \frac{1}{M}\sum_{i=1}^{M} \log P(y^{(i)}|p^{(i)}, w) - \frac{\lambda}{2}\|w\|^2 =$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left(\log \sum_{h \in H^{(i)}} \exp\left(w^\top \Phi(p^{(i)}, y^{(i)}, h)\right) - \log Z(w, i)\right) - \frac{\lambda}{2}\|w\|^2$$

where:

$$Z(w, i) = \sum_{y \in \{-1,1\}} \sum_{h \in H^i} \exp(w^\top \Phi(p_i, y, h)).$$

We maximize L using the BFGS algorithm (Nocedal and Wright, 1999). The gradient (with respect to w) is the following:

$$\nabla L = \mathbb{E}_h[\Phi(p_i, y_i, h)] - \mathbb{E}_{h,y}[\Phi(p_i, y, h)] - \lambda \cdot w$$

H_p can be defined to be any sub-set of $2^{W(p)}$ given that taking an expectation over H can be done efficiently. It is therefore possible to use prior linguistic knowledge to consider only sub-sets of p that are likely to be non-compositional (e.g., verb-preposition or verb-noun pairs).

In our experiments we attempt to keep the approach maximally general, and define H_p to be the set of all subsets of size 1 or 2 of content words in W_p[1]. We bound the size of $h \in H_p$ in order to retain computational efficiency and a sufficient frequency of the LCs in H_p. MWPs of length greater than 2 are effectively approximated by their set of subsets of sizes 1 and 2.

Each h can therefore be written as a 4-tuple $(h_L^A, h_L^B, h_R^A, h_R^B)$, where h_L^A (h_R^A) denotes the first word of the LHS (RHS) predicate's LC. h_L^B (h_R^B) denotes the (possibly empty) second word of the predicate. Inference is carried out by maximizing $P(y|p^{(i)})$ over y. As $|H_p| = O(k^4)$, where k is the number of content words in p, and as the number of content words is usually small[2], inference can be carried out by directly summing over $H^{(i)}$.

Initialization. The introduction of latent variables into the log-linear model leads to a non-convex objective function. Consequently, BFGS is not guaranteed to converge to the global optimum, but rather to a stationary point. The result may therefore depend on the parameter initialization. Indeed, preliminary experiments showed that both initializing w to be zero and using a random initializer results in lower performance.

Instead, we initialize our model with a simplified convex model that fixes the LCs to be the pair of left-most content words comprising each of the predicates. This is a common method for selecting the predicate's LC (e.g., Melamud et al., 2013a). Once h has been fixed, the model collapses to a convex log-linear model. The optimal w is then taken as an initialization point for the latent variable model. While this method may still not converge to the global maximum, our experiments show that this initialization technique yields high quality values for w (see Section 6).

3.3 Feature Set

This section lists the features used for our experiments. We intentionally select a feature set that relies on either completely unsupervised or shallow processing tools that are available for a wide variety of languages and domains.

Given a predicate pair $p^{(i)}$, a label $y \in \{1, -1\}$ and a latent state $h \in H^{(i)}$, we define their feature vector as $\Phi(p^{(i)}, y, h) = y \cdot \Phi(p^{(i)}, h)$. The computation of $\Phi(p^{(i)}, h)$ requires a reference corpus $\mathcal{R}$ that contains triplets of the type (p, x, y) where p is a binary predicate and x and y are its arguments. We use the Reverb corpus as $\mathcal{R}$ in our experiments (Fader et al., 2011; see Section 4). We refrain from encoding features that directly reflect the vocabulary of the training set. Such features are not applicable beyond that set's vocabulary, and as available datasets contain no more than a few thousand examples, these features are unlikely to generalize well.

Table 1 presents the set of features we use in our experiments. The features can be divided into two main categories: similarity features between the LHS and the RHS predicates (table's top), and features that reflect the individual properties of each

[1] We use a POS tagger to identify content words. Prepositions are considered content words under this definition.

[2] $|H_p|$ is about 15 on average in our dataset, where less than 5% of the $H^{(i)}$ are of size greater than 50.

Category		Name	Description
	Similarity	COSINE	DIRT cosine similarity between the vectors of h_L and h_R
		$\mathrm{COSINE_A}$	DIRT cosine similarity between the vectors of h_L^A and h_R^A
		BInc	DIRT BInc similarity between the vectors of h_L and h_R
		$\mathrm{BInc_A}$	DIRT BInc similarity between the vectors of h_L^A and h_R^A
	Word A LHS	$\mathrm{POS_L^A}$	The most frequent POS tag for the lemma of h_L^A
		$\mathrm{POS2_L^A}$	The second most frequent POS tag for the word lemma of h_L^A
		$\mathrm{FREQ_L^A}$	The number of occurrences of h_L^A in the reference corpus
		$\mathrm{COMMON_L^A}$	A binary feature indicating whether h_L^A appears in both predicates
		$\mathrm{ORDINAL_L^A}$	The ordinal number of h_L^A among the content words of the LHS predicate
	Pair LHS	$\mathrm{POS_L^{AB}}$	The conjunction of POS_L^A and POS_L^B
		$\mathrm{FREQ_L^{AB}}$	The frequency of h_L^A and h_L^B in the reference corpus
		$\mathrm{PREFAB_L}$	$P(h_L^A \mid h_L^A)$ as estimated from the reference corpus
		$\mathrm{PREFBA_L}$	$P(h_L^B \mid h_L^A)$ as estimated from the reference corpus
		$\mathrm{PMIAB_L}$	The point-wise mutual information of h_L^A and h_L^B
	LDA	$\mathrm{TOPICS_L}$	$P(\mathrm{topic} \mid h_L)$ for each of the induced topics.
		$\mathrm{TOPICENT_L}$	The entropy of the topic distribution $P(\mathrm{topic} \mid h_L)$

Table 1: The feature set used in our experiments. The top part presents the similarity measures based on the DIRT approach. The rest of the listed features apply to the LHS predicate (h_L), and to the first word in it (h_L^A). Analogous features are introduced for the second word, h_L^B, and for the RHS predicate. The upper-middle part presents the word features for h_L^A. The lower-middle part presents features that apply where h_L is of size 2. The bottom part lists the LDA-based features.

of them. Within the LHS feature set, we distinguish between two sub-types of features: word features that encode the individual properties of h_L^A and h_L^B (table's upper middle part), and pair features that only apply to LCs of size 2 and reflect the relation between h_L^A and h_L^B (table's lower middle part). We further incorporate LDA-based features that reflect the selectional preferences of the predicates (table's bottom).

Distributional Similarity Features. The distributional similarity features are based on the DIRT system (Lin and Pantel, 2001). The score defines for each predicate p and for each argument slot $s \in \{L, R\}$ (corresponding to the arguments to the right and left of that predicate) a vector v_s^p which represents the distribution of arguments appearing in that slot. We take $v_s^p(x)$ to be the number of times that the argument x appeared in the slot s of the predicate p. Given these vectors, the similarity between the predicates p_1 and p_2 is defined as:

$$\mathrm{score}(p_1, p_2) = \sqrt{\mathrm{sim}(v_L^{p_1}, v_L^{p_2}) \cdot \mathrm{sim}(v_R^{p_1}, v_R^{p_2})}$$

where sim is some vector similarity measure.

We use two common similarity measures: the vector cosine metric, and the BInc (Szpektor and Dagan, 2008) similarity measure. These measures give complementary perspectives on the similarity between the predicates, as the cosine similarity is symmetric between the LHS and RHS predicates, while BInc takes into account the directionality of the inference relation. Preliminary experiments with other measures, such as those of Lin (1998) and Weeds and Weir (2003) did not yield additional improvements.

We encode the similarity of all measures for the pair h_L and h_R as well as the pair h_L^A and h_R^A. The latter feature is an approximation to the similarity between the heads of the predicates, as heads in English tend to be to the left of the predicates. These two features coincide for h values of size 1.

Word and Pair Features. These features encode the basic properties of the LC. The motivation behind them is to allow a more accurate leveraging of the similarity features, as well as to better determine the relative weights of $h \in H^{(i)}$.

The feature set is composed of four analogous sets corresponding to h_L^A,h_L^B,h_R^A and h_R^B, as well as two sets of features that capture relations between h_L^A, h_L^B and h_R^A, h_R^B (in cases h is of size 2). The features include the ordinal index of the word within the predicate, the lemma's frequency according to $\mathcal{R}$, and a feature that indicates whether that word's lemma also appears in both predicates of the pair. For instance, when considering the predicates "likely to come" and "likely to leave", "likely" appears in both predicates, while "come" and "leave" appear only in one of them.

In addition, we use POS-based features that encode the most frequent POS tag for the word lemma and the second most frequent POS tag (according to $\mathcal{R}$). Information about the second most frequent POS tag can be important in identifying light verb constructions, such as "take a swim" or "give a smile", where the object is derived from a verb. It can thus be interpreted as a generalization

of the feature that indicates whether the object is a deverbal noun, which is used by some light verb identification algorithms (Tu and Roth, 2011).

In cases where h_L is of size 2, we additionally encode features that apply to the conjunction of h_L^A and h_L^B. We encode the conjunction of their POS and the number of times the two lemmas occurred together in $\mathcal{R}$. We also introduce features that capture the statistical correlation between the words of h_L. To do so, we use point-wise mutual information, and the conditional probabilities $P(h_L^A|h_L^B)$ and $P(h_L^B|h_L^A)$. Similar measures have often been used for the unsupervised detection of MWEs (Villavicencio et al., 2007; Fazly and Stevenson, 2006). We also include the analogous set of features for h_R.

LDA-based Features. We further incorporate features based on a Latent Dirichlet Allocation (LDA) topic model (Blei et al., 2003). Several recent works have underscored the usefulness of using topic models to model a predicate's selectional preferences (Ritter et al., 2010; Dinu and Lapata, 2010; Séaghdha, 2010; Lewis and Steedman, 2013; Melamud et al., 2013a). We adopt the approach of Lewis and Steedman (2013), and define a pseudo-document for each LC in the evaluation corpus. We populate the pseudo-documents of an LC with its arguments according to $\mathcal{R}$. We then train an LDA model with 25 topics over these documents. This yields a probability distribution $P(\text{topic}|h)$ for each LC h, reflecting the types of arguments h may take.

We further include a feature for the entropy of the topic distribution of the predicate, which reflects its heterogeneity. This feature is motivated by the assumption that a heterogeneous predicate is more likely to benefit from selecting a more inclusive LC than a homogeneous one.

Technical Issues. All features used, except the similarity ones and the topic distribution features are binary. Frequency features are binned into 4 bins of equal frequency. We conjoin some of the feature sets by multiplying their values. Specifically, we add the cross product of the features of the category "Similarity" (see Table 1) with the rest of the features. In addition, we conjoin all LHS (RHS) features with an indicator feature that indicates whether h_L (h_R) is of size two. This results in 1605 non-constant features.

We further note that some LCs that appear in the evaluation corpus do not appear at all in $\mathcal{R}$. In our experiments they amounted to 0.2% of the LCs in our evaluation dataset. While previous work often discarded predicates below a certain frequency from the evaluation, we include them in order to facilitate comparison to future work. We assign the similarity features of such examples a 0 value, and assign their other numerical features the mean value of those features.

4 Experimental Setup

Corpora and Preprocessing. As a reference corpus $\mathcal{R}$, we use Reverb (Fader et al., 2011), a web-based corpus consisting of 15M web extractions of binary relations. Each relation is a triplet of a predicate and two arguments, one preceding it and one following it. Relations were extracted using regular expressions over the output of a POS tagger and an NP chunker. Each predicate may consist of a single verb, a verb and a preposition or a sequence of words starting in a verb and ending in a preposition, between which there may nouns, adjectives, adverbs, pronouns, determiners and verbs. The verb may also be a copula. Examples of predicates are "make the most of", "could be exchanged for" and "is happy with".

Reverb is an appealing reference corpus for this task for several reasons. First, it uses fairly shallow preprocessing technology which is available for many domains and languages. Second, Reverb applies considerable noise filtering, which results in extractions of fair quality. Third, our evaluation dataset is based on Reverb extractions.

We evaluate our algorithm on the dataset of Zeichner et al. (2012). This publicly available corpus[3] provides pairs of Reverb binary relations and an indication of whether an inference relation holds between them within the context of a specific pair of argument fillers. The corpus was compiled using distributional methods to detect pairs of relations in Reverb that are likely to have an inference relation between. Annotators, employed through Amazon Mechanical Turk, were then asked to determine whether each pair is meaningful, and if so, to determine whether an inference relation holds. Further measures were taken to monitor the accuracy of the annotation.

For example, the pair of predicates "make the most of" and "take advantage of" appears in the corpus as a pair between which an inference relation holds. The arguments in this case are "students" and "their university experience". An ex-

[3] http://tinyurl.com/krx2acd

ample of a pair between which an inference relation does not hold is "tend to neglect" and "underestimate the importance of", where the arguments are "Robert" and "his family".

The dataset contains 6,565 instances in total. We use 5,411 pairs of them, discarding instances that were deemed as meaningless by the annotators. We also discard cases where the set of arguments is reversed between the LHS and RHS predicates. In these examples, $p_R(x, y)$ is inferable from $p_L(y, x)$, rather than from $p_L(x, y)$. As there are less than 150 reversed instances in the corpus, experimenting on this sub-set is unlikely to be informative.

The average length of a predicate in the corpus is 2.7 words (including function words). In 87.3% of the predicate pairs, there was more than one LC (i.e., $|H_p| > 1$), underscoring the importance of correctly leveraging the different LCs. We randomly partition the corpus into a training set which contains 4,343 instances (∼80%), and a test set that contains 1,068 instances, maintaining the same positive to negative label ratio in both datasets[4]. Development was carried out using cross-validation on the training data (see below).

We use a Maximum Entropy POS Tagger, trained on the Penn Treebank, and the WordNet lemmatizer, both implemented within the NLTK package (Loper and Bird, 2002). To obtain a coarse-grained set of POS tags, we collapse the tag set to 7 categories: nouns, verbs, adjectives, adverbs, prepositions, the word "to" and a category that includes all other words. A Reverb argument is represented as the conjunction of its content words that appear more than 10 times in the corpus. Function words are defined according to their POS tags and include determiners, possessive pronouns, existential "there", numbers and coordinating conjunctions. Auxiliary verbs and copulas are also considered function words.

To compute the LDA features, we use the online variational Bayes algorithm of (Hoffman et al., 2010) as implemented in the *Gensim* software package (Rehurek and Sojka, 2010).

Evaluated Algorithms. The only two previous works on this dataset (Melamud et al., 2013a; Melamud et al., 2013b) are not directly comparable, as they used unsupervised systems and evaluated on sub-sets of the evaluation dataset. Instead, we use several baselines to demonstrate the usefulness of integrating multiple LCs, as well as the relative usefulness of our feature sets.

The simplest baseline is ALLNEG, which predicts the most frequent label in the dataset (in our case: "no inference"). The other evaluated systems are formed by taking various subsets of our feature set. We experiment with 4 feature sets. The smallest set, SIM, includes only the similarity features. This feature set is related to the compositional distributional model of Mitchell and Lapata (2010) (see Section 6). We note that despite recent advances in identifying predicate inference relations, the DIRT system (Lin and Pantel, 2001) remains a strong baseline, and is often used as a component in state-of-the-art systems (Berant et al., 2011), and specifically in the two aforementioned works that used the same evaluation corpus.

The next feature set BASIC includes the features found to be most useful during the development of the model: the most frequent POS tag, the frequency features and the feature Common. More inclusive is the feature set NO-LDA, which includes all features except the LDA features. Experiments with this set were performed in order to isolate the effect of the LDA features. Finally, ALL includes our complete set of features.

The more direct comparison is against partial implementations of our system where the LC h is deterministically selected. Determining h for each predicate yields a regular log-linear binary classification model. We use two variants of this baseline. The first, LEFTMOST, selects the left-most content word for each predicate. Similar selection strategy was carried out by Melamud et al. (2013a). The second, VPREP, selects h to be the verb along with its following preposition. In cases the predicate contains multiple verbs, the one preceding the preposition is selected, and where the predicate does not contain any non-copula verbs, it regresses to LEFTMOST. This LC selection method approximates a baseline that includes subcategorized prepositions. Such cases are highly frequent and account for a large portion of the MWPs in English. Including a verb's preposition in its LC was commonly done in previous work (e.g., Lewis and Steedman, 2013).

We also attempted to identify verb-preposition constructions using a dependency parser. Unfortunately, our evaluation dataset is only available in

[4]A script that replicates our train-test partition of the corpus can be found here: `http://homepages.inf.ed.ac.uk/oabend/mwpreds.html`

a lemmatized version, which posed a difficulty for the parser. Due to the low quality of the resulting parses, we implemented VPREP using POS-based regular expressions as defined above.

The full model is denoted with LATENTLC. For each system and feature set, we report results using 10-fold cross-validation on the training set, as well as results on the test set. Both cases use the same set of parameters determined by cross-validation on the training set. As the task at hand is a binary classification problem, we use accuracy scores to rate the performance of our systems.

5 Results

Table 2 presents the results of our experiments. Rows correspond to the evaluated algorithms, while columns correspond to the feature sets used and the evaluation scenarios (i.e., training set cross-validation or test set evaluation). Our experiments make first use of this dataset in its fullest form for the problem of supervised learning of inference relations, and may serve as a starting point for further exploration of this dataset.

For all feature sets and settings, LATENTLC scored highest, often with a considerable margin of up to 3.0% in the cross-validation and up to 4.6% on the test set relative to the LEFTMOST baseline, and 5.1% (cross-validation) and 6.8% (test) margins relative to VPREP.

The best scoring result of our LATENTLC model in the cross-validation scenario is 65.72%, obtained by the feature set All. The best scoring result by any of the baseline models in this scenario is 62.7%, obtained by the same feature set. For the test set scenario, LATENTLC obtained its highest accuracy, 65.73%, when using the feature set Basic. This is a substantial improvement over the highest scoring baseline model in this scenario that obtained 61.6% accuracy, using the feature set All. This performance gap is substantial when taking into consideration that the improvements obtained by the highly competitive DIRT similarity features using the stronger LEFTMOST baseline, result in an improvement of 3.1% and 5.3% over the trivial ALLNEG baseline in the test set and cross-validation scenarios respectively.

Comparing the different feature sets on our proposed model, we find that the Basic feature set gives a consistent and substantial increase over the Sim feature set. Improvements are of 2.8% (test) and 2.2% (cross-validation). Introducing more elaborate features (i.e., the feature sets NoLDA and All) yields some improvements in the cross-validation, but these improvements are not replicated on the test set. This may be due to idiosyncrasies in the test set that are averaged out in the cross-validation scenario.

For a qualitative analysis, we took the best performing model of the data set (i.e., with the Basic feature set), and extracted the set of instances where it made a correct prediction while both baselines made an error. This set contains many verb-preposition pairs, such as "list as → report as" or "submit via → deliver by", underscoring the utility of leveraging multiple LCs rather than considering only a head word (as with LEFTMOST) or the entire phrase (as with VPREP). Other examples in this set contain more complex patterns. These include the positive pairs "talk much about → have much to say about" and "increase with → go up with", and the negative "make prediction about → meet the challenge of" and "enjoy watching → love to play".

6 Discussion

Relation to CDS. Much recent work subsumed under the title *Compositional Distributional Semantics* addressed the distributional representation of multi-word phrases (see Section 2). This line of work focuses on compositional predicates, such as "kick the ball" and not on idiosyncratic predicates such as "kick the bucket".

A variant of the CDS approach can be framed within ours. Assume we wish to compute the similarity of the predicates $p_L = (w_1, ..., w_n)$ and $p_R = (w'_1, ..., w'_m)$. Let us denote the vector space representations of the individual words as $v_1, ..., v_n$ and $v'_1, ..., v'_m$ respectively. A standard approach in CDS is to compose distributional representations by taking their vector sum $v_L = v_1 + v_2 ... + v_n$ and $v_R = v'_1 + ... + v'_m$ (Mitchell and Lapata, 2010). One of the most effective similarity measures is the cosine similarity, which is a normalized dot product. The distributional similarity between p_L and p_R under this model is $\mathrm{sim}(p_L, p_R) = \sum_{i=1}^{n} \sum_{j=1}^{m} \mathrm{sim}(w_i, w'_j)$, where $\mathrm{sim}(w_i, w'_j)$ is the dot product between v_i and v'_j.

This similarity score is similar in spirit to a simplified version of our statistical model that restricts the set of allowable LCs H_p to be $\{(\{w_i\}, \{w'_j\}) | i \leq n, j \leq m\}$, i.e., only LCs of size 1. Indeed, taking H_p as above, and cosine similarity as the only feature (i.e., $w \in \mathbb{R}$), yields the distribution

Algorithm	Test Set				Cross Validation			
	Sim	Basic	NoLDA	All	Sim	Basic	NoLDA	All
LatentLC	**62.9**	**65.7**	**64.4**	**64.6**	**62.7** ± 1.9	**64.9** ± 1.9	**65.0** ± 1.7	**65.7** ±1.9
LeftMost	59.0	61.1	60.0	60.4	61.2 ± 2.1	62.5 ± 2.4	62.4 ±2.2	62.7 ± 2.0
VPrep	56.1	60.9	60.7	61.6*	58.1 ± 1.7	60.8 ± 2.2	60.4 ± 2.6	60.6 ± 2.2
AllNeg	55.9				55.9			

Table 2: Results for the various evaluated systems. Accuracy results are presented in percents, followed in the cross validation scenario by the standard deviation over the folds. The rows correspond to the various systems as defined in Section 4. LatentLC is our proposed model. The columns correspond to the various feature sets, from the least to the most inclusive. Sim includes only similarity features. Basic additionally includes POS-based and frequency features. NoLDA includes all features except LDA-based features. All is the full feature set. AllNeg is the classifier that invariably predicts the label "no inference". Bold marks best overall accuracy per column, and * marks figures that are not significantly worse (McNemar's test, $p < 0.05$). The same positive to negative label ratio was maintained in both the cross validation and test set scenarios. In all cases, LatentLC obtains substantial improvements over the baseline systems.

$$P(y|p) \propto \sum_{(w_i,w'_j)\in H_p} \exp\left(w \cdot y \cdot \mathrm{sim}(w_i, w'_j)\right).$$

This derivation highlights the relation of a simplified version of our approach to the additive CDS model, as both approaches effectively average over the similarities of all pairs of words in p_L and p_R. The derivation also highlights a few advantages of our approach. First, our approach allows to straightforwardly introduce additional features and to weight them in a way most consistent with the task at hand. Second, it allows much more flexibility in defining the set of allowable LCs, H_p. Specifically, H_p may contain LCs of sizes greater than 1. Third, our approach uses standard probabilistic modelling, and therefore has a natural statistical interpretation.

In order to appreciate the effect of these advantages, we perform an experiment that takes H to be the set of all LCs of size 1, and uses a single similarity measure. We run a 10-fold cross-validation on our training data, obtaining 61.3% accuracy using Cosine and 62.2% accuracy using BInc. The performance gap between these results and the accuracy obtained by our full model (65.7%) underscores the latter's effectiveness in integrating multiple features and LCs.

Effectiveness of Optimization Method. Our maximization of the log-likelihood function is not guaranteed to converge to a global optimum. Therefore, the quality of the learned parameters may be sensitive to the initialization point. We hereby describe an experiment that tests the sensitivity of our approach to such variance.

Selecting the highest scoring feature set on our test set (i.e., Basic), we ran the model with multiple initializers, by randomly perturbing our standard convex initializer (see Section 3). Concretely, given a convex initializer w, we select the starting point to be $w + \eta$, where $\eta_i \sim \mathcal{N}(0, \alpha|w_i|)$. We ran this experiment 400 times with $\alpha = 0.8$.

To combine the resulting weight vectors into a single classifier, we apply two types of standard approaches: a Product of Experts (Hinton, 2002), as well as a voting approach that selects the most frequently predicted label. Neither of these experiments yielded any significant performance gain. This demonstrates the robustness of our optimization method to the initialization point.

7 Conclusion

We have presented a novel approach to the distributional representation of multi-word predicates. Since MWPs demonstrate varying levels of compositionality, a uniform treatment of MWPs either as fixed expressions or through head words is lacking. Instead, our approach integrates multiple lexical units contained in the predicate. The approach takes into account both multi-word LCs that address low compositionality cases, as well as single-word LCs that address compositional cases and are more frequent. It assumes a latent distribution over the LCs of the predicates, and estimates it relative to a target application task.

We addressed the supervised inference identification task, obtaining substantial improvement over state-of-the-art baseline systems. In future work we intend to assess the benefit of this approach in MWP classes that are well-known from the literature. We believe that a permissive approach that integrates multiple analyses would perform better than standard single-analysis methods in a wide range of applications.

Acknowledgements. We would like to thank Mike Lewis, Reshef Meir, Oren Melamud, Michael Roth and Nathan Schneider for their helpful comments. This work was supported by ERC Advanced Fellowship 249520 GRAMPLUS.

References

Alex Alsina, Joan Wanda Bresnan, and Peter Sells. 1997. *Complex predicates*. Center for the Study of Language and Information.

Timothy Baldwin, Colin Bannard, Takaaki Tanaka, and Dominic Widdows. 2003. An empirical model of multiword expression decomposability. In *Proceedings of the ACL 2003 workshop on Multiword expressions: analysis, acquisition and treatment-Volume 18*, pages 89–96.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *EMNLP*, pages 1183–1193.

Jonathan Berant, Jacob Goldberger, and Ido Dagan. 2011. Global learning of typed entailment rules. In *ACL*, pages 610–619.

Chris Biemann and Eugenie Giesbrecht. 2011. Distributional semantics and compositionality 2011: Shared task description and results. In *Workshop on Distributional Semantics and Compositionality*, pages 21–28.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent Dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Miriam Butt. 2010. The light verb jungle: still hacking away. In *Complex predicates: cross-linguistic perspectives on event structure*, pages 48–78. Cambridge University Press.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. In J. van Bentham, M. Moortgat, and W. Buszkowski, editors, *Linguistic Analysis*, volume 36, pages 435–384.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. Recognizing textual entailment: Models and applications. *Synthesis Lectures on Human Language Technologies*, 6(4):1–220.

Georgiana Dinu and Mirella Lapata. 2010. Topic models for meaning similarity in context. In *COLING: Posters*, pages 250–258.

Georgiana Dinu and Rui Wang. 2009. Inference rules and their application to recognizing textual entailment. In *EACL*, pages 211–219.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *EMNLP*, pages 1535–1545.

Afsaneh Fazly and Suzanne Stevenson. 2006. Automatically constructing a lexicon of verb phrase idiomatic combinations. In *EACL*, pages 337–344.

Masato Hagiwara, Yasuhiro Ogawa, and Katsuhiko Toyama. 2009. Supervised synonym acquisition using distributional features and syntactic patterns. *Information and Media Technologies*, 4(2):558–582.

Geoffrey E Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800.

Matthew Hoffman, Francis R Bach, and David M Blei. 2010. Online learning for latent Dirichlet allocation. In *NIPS*, pages 856–864.

Ray Jackendoff. 2002. *Foundations of language: Brain, meaning, grammar, evolution*. Oxford University Press.

Douwe Kiela and Stephen Clark. 2013. Detecting compositionality of multi-word expressions using nearest neighbours in vector space models. In *EMNLP*, pages 1427–1432.

Mike Lewis and Mark Steedman. 2013. Combined distributional and logical semantics. *TACL*, 1:179–192.

Dekang Lin and Patrick Pantel. 2001. DIRT – discovery of inference rules from text. In *SIGKDD 2001*, pages 323–328.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *COLING-ACL*, pages 768–774.

Dekang Lin. 1999. Automatic identification of non-compositional phrases. In *ACL*, pages 317–324.

Edward Loper and Steven Bird. 2002. NLTK: The natural language toolkit. In *ACL Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics*, pages 63–70.

Diana McCarthy, Bill Keller, and John Carroll. 2003. Detecting a continuum of compositionality in phrasal verbs. In *ACL workshop on Multiword expressions: analysis, acquisition and treatment*, pages 73–80.

Oren Melamud, Jonathan Berant, Ido Dagan, Jacob Goldberger, and Idan Szpektor. 2013a. A two level model for context sensitive inference rules. In *ACL 2013*, pages 1331–1340.

Oren Melamud, Ido Dagan, Jacob Goldberger, and Idan Szpektor. 2013b. Using lexical expansion to learn inference rules from sparse data. In *ACL: Short Papers*, pages 283–288.

Shachar Mirkin, Ido Dagan, and Maayan Geffet. 2006. Integrating pattern-based and distributional similarity methods for lexical entailment acquisition. In *COLING-ACL: Poster Session*, pages 579–586.

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

Jorge. Nocedal and Stephen J Wright. 1999. *Numerical optimization*, volume 2. Springer New York.

Karl Pichotta and John DeNero. 2013. Identifying phrasal verbs using many bilingual corpora. In *EMNLP*, pages 636–646.

Carlos Ramisch, Aline Villavicencio, and Valia Kordoni. 2013. Introduction to the special issue on multiword expressions: From theory to practice and use. *ACM Transactions on Speech and Language Processing (TSLP)*, 10(2):3.

Deepak Ravichandran and Eduard Hovy. 2002. Learning surface text patterns for a question answering system. In *ACL*, pages 41–47.

Siva Reddy, Diana McCarthy, and Suresh Manandhar. 2011. An empirical study on compositionality in compound nouns. In *IJCNLP*, pages 210–218.

Radim Rehurek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *Proceedings of LREC 2010 workshop New Challenges for NLP Frameworks*, pages 46–50.

Alan Ritter, Mausam, and Oren Etzioni. 2010. A latent Dirichlet allocation method for selectional preferences. In *ACL*, pages 424–434.

Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2001. Multiword expressions: A pain in the neck for NLP. In *CICLing*, pages 1–15.

Stefan Schoenmackers, Oren Etzioni, Daniel S Weld, and Jesse Davis. 2010. Learning first-order Horn clauses from web text. In *EMNLP*, pages 1088–1098.

Diarmuid Ó. Séaghdha. 2010. Latent variable models of selectional preference. In *ACL 2010*, pages 435–444.

Maggie Seaton and Alison Macaulay, editors. 2002. *Collins COBUILD Idioms Dictionary*. HarperCollins Publishers, 2nd edition.

Satoshi Sekine. 2005. Automatic paraphrase discovery based on context and keywords between NE pairs. In *IWP*, pages 4–6.

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

Idan Szpektor and Ido Dagan. 2008. Learning entailment rules for unary templates. In *COLING*, pages 849–856.

Yuancheng Tu and Dan Roth. 2011. Learning English light verb constructions: contextual or statistical. In *ACL HLT 2011*, page 31.

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37(1):141–188.

Aline Villavicencio, Valia Kordoni, Yi Zhang, Marco Idiart, and Carlos Ramisch. 2007. Validation and evaluation of automatically acquired multiword expressions for grammar engineering. In *EMNLP-CoNLL*, pages 1034–1043.

Veronika Vincze, István Nagy T., and Richárd Farkas. 2013. Identifying English and Hungarian light verb constructions: A contrastive approach. In *ACL: Short Papers*, pages 255–261.

Julie Weeds and David Weir. 2003. A general framework for distributional similarity. In *EMNLP*, pages 81–88.

Hila Weisman, Jonathan Berant, Idan Szpektor, and Ido Dagan. 2012. Learning verb inference rules from linguistically-motivated evidence. In *EMNLP-CoNLL*, pages 194–204.

Dominic Widdows. 2008. Semantic vector products: Some initial investigations. In *Second AAAI Symposium on Quantum Interaction*, volume 26, pages 28–35.

Alison Wray. 2008. *Formulaic language: Pushing the boundaries*. Oxford University Press.

Fabio Massimo Zanzotto, Marco Pennacchiotti, and Maria Teresa Pazienza. 2006. Discovering asymmetric entailment relations between verbs using selectional preferences. In *ACL-COLING*, pages 849–856.

Naomi Zeichner, Jonathan Berant, and Ido Dagan. 2012. Crowdsourcing inference-rule evaluation. In *ACL: Short Papers*, pages 156–160.

A Convolutional Neural Network for Modelling Sentences

Nal Kalchbrenner **Edward Grefenstette** **Phil Blunsom**

{nal.kalchbrenner, edward.grefenstette, phil.blunsom}@cs.ox.ac.uk
Department of Computer Science
University of Oxford

Abstract

The ability to accurately represent sentences is central to language understanding. We describe a convolutional architecture dubbed the Dynamic Convolutional Neural Network (DCNN) that we adopt for the semantic modelling of sentences. The network uses Dynamic k-Max Pooling, a global pooling operation over linear sequences. The network handles input sentences of varying length and induces a feature graph over the sentence that is capable of explicitly capturing short and long-range relations. The network does not rely on a parse tree and is easily applicable to any language. We test the DCNN in four experiments: small scale binary and multi-class sentiment prediction, six-way question classification and Twitter sentiment prediction by distant supervision. The network achieves excellent performance in the first three tasks and a greater than 25% error reduction in the last task with respect to the strongest baseline.

1 Introduction

The aim of a sentence model is to analyse and represent the semantic content of a sentence for purposes of classification or generation. The sentence modelling problem is at the core of many tasks involving a degree of natural language comprehension. These tasks include sentiment analysis, paraphrase detection, entailment recognition, summarisation, discourse analysis, machine translation, grounded language learning and image retrieval. Since individual sentences are rarely observed or not observed at all, one must represent a sentence in terms of features that depend on the words and short n-grams in the sentence that are frequently observed. The core of a sentence model involves a feature function that defines the process by which the features of the sentence are extracted from the features of the words or n-grams.

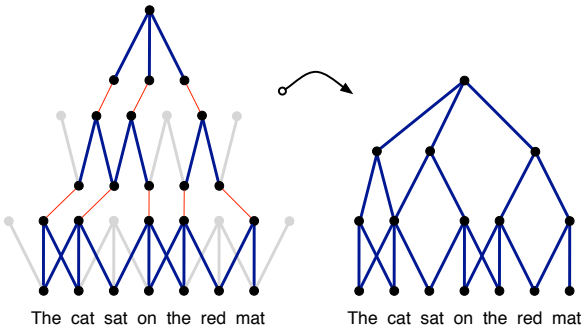


Figure 1: Subgraph of a feature graph induced over an input sentence in a Dynamic Convolutional Neural Network. The full induced graph has multiple subgraphs of this kind with a distinct set of edges; subgraphs may merge at different layers. The left diagram emphasises the pooled nodes. The width of the convolutional filters is 3 and 2 respectively. With dynamic pooling, a filter with small width at the higher layers can relate phrases far apart in the input sentence.

Various types of models of meaning have been proposed. Composition based methods have been applied to vector representations of word meaning obtained from co-occurrence statistics to obtain vectors for longer phrases. In some cases, composition is defined by algebraic operations over word meaning vectors to produce sentence meaning vectors (Erk and Padó, 2008; Mitchell and Lapata, 2008; Mitchell and Lapata, 2010; Turney, 2012; Erk, 2012; Clarke, 2012). In other cases, a composition function is learned and either tied to particular syntactic relations (Guevara, 2010; Zanzotto et al., 2010) or to particular word types (Baroni and Zamparelli, 2010; Coecke et al., 2010; Grefenstette and Sadrzadeh, 2011; Kartsaklis and Sadrzadeh, 2013; Grefenstette, 2013). Another approach represents the meaning of sentences by way of automatically extracted logical forms (Zettlemoyer and Collins, 2005).

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 655–665,
Baltimore, Maryland, USA, June 23-25 2014.

A central class of models are those based on neural networks. These range from basic neural bag-of-words or bag-of-n-grams models to the more structured recursive neural networks and to time-delay neural networks based on convolutional operations (Collobert and Weston, 2008; Socher et al., 2011; Kalchbrenner and Blunsom, 2013b). Neural sentence models have a number of advantages. They can be trained to obtain generic vectors for words and phrases by predicting, for instance, the contexts in which the words and phrases occur. Through supervised training, neural sentence models can fine-tune these vectors to information that is specific to a certain task. Besides comprising powerful classifiers as part of their architecture, neural sentence models can be used to condition a neural language model to generate sentences word by word (Schwenk, 2012; Mikolov and Zweig, 2012; Kalchbrenner and Blunsom, 2013a).

We define a convolutional neural network architecture and apply it to the semantic modelling of sentences. The network handles input sequences of varying length. The layers in the network interleave one-dimensional convolutional layers and dynamic k-max pooling layers. Dynamic k-max pooling is a generalisation of the max pooling operator. The max pooling operator is a non-linear subsampling function that returns the maximum of a set of values (LeCun et al., 1998). The operator is generalised in two respects. First, k-max pooling over a linear sequence of values returns the subsequence of k maximum values in the sequence, instead of the single maximum value. Secondly, the pooling parameter k can be dynamically chosen by making k a function of other aspects of the network or the input.

The convolutional layers apply one-dimensional filters across each row of features in the sentence matrix. Convolving the same filter with the n-gram at every position in the sentence allows the features to be extracted independently of their position in the sentence. A convolutional layer followed by a dynamic pooling layer and a non-linearity form a feature map. Like in the convolutional networks for object recognition (LeCun et al., 1998), we enrich the representation in the first layer by computing multiple feature maps with different filters applied to the input sentence. Subsequent layers also have multiple feature maps computed by convolving filters with all the maps from the layer below. The weights at these layers form an order-4 tensor. The resulting architecture is dubbed a Dynamic Convolutional Neural Network.

Multiple layers of convolutional and dynamic pooling operations induce a structured feature graph over the input sentence. Figure 1 illustrates such a graph. Small filters at higher layers can capture syntactic or semantic relations between non-continuous phrases that are far apart in the input sentence. The feature graph induces a hierarchical structure somewhat akin to that in a syntactic parse tree. The structure is not tied to purely syntactic relations and is internal to the neural network.

We experiment with the network in four settings. The first two experiments involve predicting the sentiment of movie reviews (Socher et al., 2013b). The network outperforms other approaches in both the binary and the multi-class experiments. The third experiment involves the categorisation of questions in six question types in the TREC dataset (Li and Roth, 2002). The network matches the accuracy of other state-of-the-art methods that are based on large sets of engineered features and hand-coded knowledge resources. The fourth experiment involves predicting the sentiment of Twitter posts using distant supervision (Go et al., 2009). The network is trained on 1.6 million tweets labelled automatically according to the emoticon that occurs in them. On the hand-labelled test set, the network achieves a greater than 25% reduction in the prediction error with respect to the strongest unigram and bigram baseline reported in Go et al. (2009).

The outline of the paper is as follows. Section 2 describes the background to the DCNN including central concepts and related neural sentence models. Section 3 defines the relevant operators and the layers of the network. Section 4 treats of the induced feature graph and other properties of the network. Section 5 discusses the experiments and inspects the learnt feature detectors.[1]

2 Background

The layers of the DCNN are formed by a convolution operation followed by a pooling operation. We begin with a review of related neural sentence models. Then we describe the operation of *one-dimensional convolution* and the classical Time-Delay Neural Network (TDNN) (Hinton, 1989; Waibel et al., 1990). By adding a max pooling

[1] Code available at `www.nal.co`

layer to the network, the TDNN can be adopted as a sentence model (Collobert and Weston, 2008).

2.1 Related Neural Sentence Models

Various neural sentence models have been described. A general class of basic sentence models is that of Neural Bag-of-Words (NBoW) models. These generally consist of a projection layer that maps words, sub-word units or n-grams to high dimensional embeddings; the latter are then combined component-wise with an operation such as summation. The resulting combined vector is classified through one or more fully connected layers.

A model that adopts a more general structure provided by an external parse tree is the Recursive Neural Network (RecNN) (Pollack, 1990; Küchler and Goller, 1996; Socher et al., 2011; Hermann and Blunsom, 2013). At every node in the tree the contexts at the left and right children of the node are combined by a classical layer. The weights of the layer are shared across all nodes in the tree. The layer computed at the top node gives a representation for the sentence. The Recurrent Neural Network (RNN) is a special case of the recursive network where the structure that is followed is a simple linear chain (Gers and Schmidhuber, 2001; Mikolov et al., 2011). The RNN is primarily used as a language model, but may also be viewed as a sentence model with a linear structure. The layer computed at the last word represents the sentence.

Finally, a further class of neural sentence models is based on the convolution operation and the TDNN architecture (Collobert and Weston, 2008; Kalchbrenner and Blunsom, 2013b). Certain concepts used in these models are central to the DCNN and we describe them next.

2.2 Convolution

The *one-dimensional convolution* is an operation between a vector of weights $\mathbf{m} \in \mathbb{R}^m$ and a vector of inputs viewed as a sequence $\mathbf{s} \in \mathbb{R}^s$. The vector $\mathbf{m}$ is the *filter* of the convolution. Concretely, we think of $\mathbf{s}$ as the input sentence and $\mathbf{s}_i \in \mathbb{R}$ is a single feature value associated with the i-th word in the sentence. The idea behind the one-dimensional convolution is to take the dot product of the vector $\mathbf{m}$ with each m-gram in the sentence $\mathbf{s}$ to obtain another sequence $\mathbf{c}$:

$$\mathbf{c}_j = \mathbf{m}^\intercal \mathbf{s}_{j-m+1:j} \tag{1}$$

Equation 1 gives rise to two types of convolution depending on the range of the index j. The *narrow* type of convolution requires that $s \geq m$ and yields

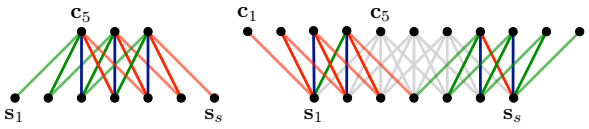


Figure 2: Narrow and wide types of convolution. The filter $\mathbf{m}$ has size $m = 5$.

a sequence $\mathbf{c} \in \mathbb{R}^{s-m+1}$ with j ranging from m to s. The *wide* type of convolution does not have requirements on s or m and yields a sequence $\mathbf{c} \in \mathbb{R}^{s+m-1}$ where the index j ranges from 1 to $s + m - 1$. Out-of-range input values $\mathbf{s}_i$ where $i < 1$ or $i > s$ are taken to be zero. The result of the narrow convolution is a subsequence of the result of the wide convolution. The two types of one-dimensional convolution are illustrated in Fig. 2.

The trained weights in the filter $\mathbf{m}$ correspond to a linguistic feature detector that learns to recognise a specific class of n-grams. These n-grams have size $n \leq m$, where m is the width of the filter. Applying the weights $\mathbf{m}$ in a wide convolution has some advantages over applying them in a narrow one. A wide convolution ensures that all weights in the filter reach the entire sentence, including the words at the margins. This is particularly significant when m is set to a relatively large value such as 8 or 10. In addition, a wide convolution guarantees that the application of the filter $\mathbf{m}$ to the input sentence $\mathbf{s}$ always produces a valid non-empty result $\mathbf{c}$, independently of the width m and the sentence length s. We next describe the classical convolutional layer of a TDNN.

2.3 Time-Delay Neural Networks

A TDNN convolves a sequence of inputs $\mathbf{s}$ with a set of weights $\mathbf{m}$. As in the TDNN for phoneme recognition (Waibel et al., 1990), the sequence $\mathbf{s}$ is viewed as having a time dimension and the convolution is applied over the time dimension. Each $\mathbf{s}_j$ is often not just a single value, but a vector of d values so that $\mathbf{s} \in \mathbb{R}^{d \times s}$. Likewise, $\mathbf{m}$ is a matrix of weights of size $d \times m$. Each row of $\mathbf{m}$ is convolved with the corresponding row of $\mathbf{s}$ and the convolution is usually of the narrow type. Multiple convolutional layers may be stacked by taking the resulting sequence $\mathbf{c}$ as input to the next layer.

The Max-TDNN sentence model is based on the architecture of a TDNN (Collobert and Weston, 2008). In the model, a convolutional layer of the narrow type is applied to the sentence matrix $\mathbf{s}$, where each column corresponds to the feature vec-

tor $\mathbf{w}_i \in \mathbb{R}^d$ of a word in the sentence:

$$\mathbf{s} = \begin{bmatrix} | & | & | \\ \mathbf{w}_1 & \dots & \mathbf{w}_s \\ | & | & | \end{bmatrix} \tag{2}$$

To address the problem of varying sentence lengths, the Max-TDNN takes the maximum of each row in the resulting matrix $\mathbf{c}$ yielding a vector of d values:

$$\mathbf{c}_{max} = \begin{bmatrix} \max(\mathbf{c}_{1,:}) \\ \vdots \\ \max(\mathbf{c}_{d,:}) \end{bmatrix} \tag{3}$$

The aim is to capture the most relevant feature, i.e. the one with the highest value, for each of the d rows of the resulting matrix $\mathbf{c}$. The fixed-sized vector $\mathbf{c}_{max}$ is then used as input to a fully connected layer for classification.

The Max-TDNN model has many desirable properties. It is sensitive to the order of the words in the sentence and it does not depend on external language-specific features such as dependency or constituency parse trees. It also gives largely uniform importance to the signal coming from each of the words in the sentence, with the exception of words at the margins that are considered fewer times in the computation of the narrow convolution. But the model also has some limiting aspects. The range of the feature detectors is limited to the span m of the weights. Increasing m or stacking multiple convolutional layers of the narrow type makes the range of the feature detectors larger; at the same time it also exacerbates the neglect of the margins of the sentence and increases the minimum size s of the input sentence required by the convolution. For this reason higher-order and long-range feature detectors cannot be easily incorporated into the model. The max pooling operation has some disadvantages too. It cannot distinguish whether a relevant feature in one of the rows occurs just one or multiple times and it forgets the order in which the features occur. More generally, the pooling factor by which the signal of the matrix is reduced at once corresponds to $s - m + 1$; even for moderate values of s the pooling factor can be excessive. The aim of the next section is to address these limitations while preserving the advantages.

3 Convolutional Neural Networks with Dynamic k-Max Pooling

We model sentences using a convolutional architecture that alternates wide convolutional layers with dynamic pooling layers given by *dynamic k-max pooling*. In the network the width of a feature map at an intermediate layer varies depending on the length of the input sentence; the resulting architecture is the Dynamic Convolutional Neural Network. Figure 3 represents a DCNN. We proceed to describe the network in detail.

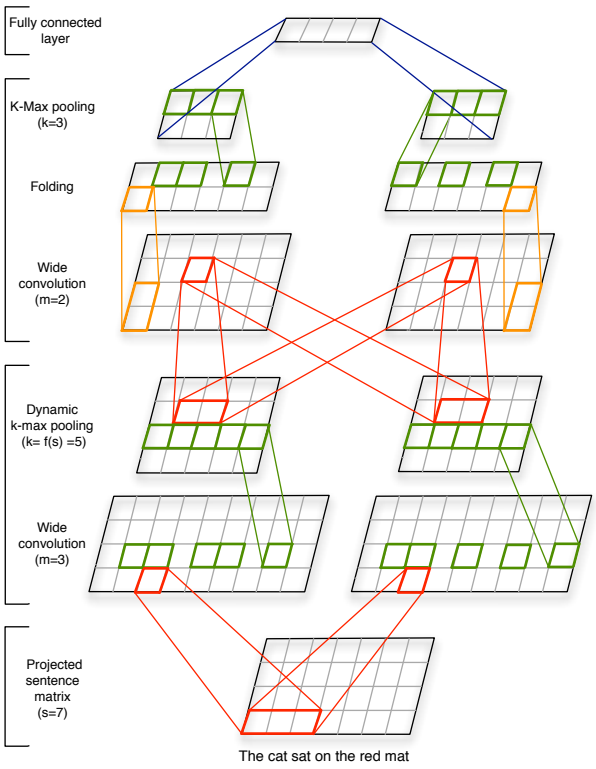


Figure 3: A DCNN for the seven word input sentence. Word embeddings have size $d = 4$. The network has two convolutional layers with two feature maps each. The widths of the filters at the two layers are respectively 3 and 2. The (dynamic) k-max pooling layers have values k of 5 and 3.

3.1 Wide Convolution

Given an input sentence, to obtain the first layer of the DCNN we take the embedding $\mathbf{w}_i \in \mathbb{R}^d$ for each word in the sentence and construct the sentence matrix $\mathbf{s} \in \mathbb{R}^{d \times s}$ as in Eq. 2. The values in the embeddings $\mathbf{w}_i$ are parameters that are optimised during training. A convolutional layer in the network is obtained by convolving a matrix of weights $\mathbf{m} \in \mathbb{R}^{d \times m}$ with the matrix of activations at the layer below. For example, the second layer is obtained by applying a convolution to the sentence matrix $\mathbf{s}$ itself. Dimension d and filter width m are hyper-parameters of the network. We let the operations be *wide* one-dimensional convolutions as described in Sect. 2.2. The resulting matrix $\mathbf{c}$ has dimensions $d \times (s + m - 1)$.

3.2 k-Max Pooling

We next describe a pooling operation that is a generalisation of the max pooling over the time dimension used in the Max-TDNN sentence model and different from the local max pooling operations applied in a convolutional network for object recognition (LeCun et al., 1998). Given a value k and a sequence $\mathbf{p} \in \mathbb{R}^p$ of length $p \geq k$, *k-max pooling* selects the subsequence $\mathbf{p}^k_{max}$ of the k highest values of $\mathbf{p}$. The order of the values in $\mathbf{p}^k_{max}$ corresponds to their original order in $\mathbf{p}$.

The k-max pooling operation makes it possible to pool the k most active features in $\mathbf{p}$ that may be a number of positions apart; it preserves the order of the features, but is insensitive to their specific positions. It can also discern more finely the number of times the feature is highly activated in $\mathbf{p}$ and the progression by which the high activations of the feature change across $\mathbf{p}$. The k-max pooling operator is applied in the network after the topmost convolutional layer. This guarantees that the input to the fully connected layers is independent of the length of the input sentence. But, as we see next, at intermediate convolutional layers the pooling parameter k is not fixed, but is dynamically selected in order to allow for a smooth extraction of higher-order and longer-range features.

3.3 Dynamic k-Max Pooling

A *dynamic k-max pooling* operation is a k-max pooling operation where we let k be a function of the length of the sentence and the depth of the network. Although many functions are possible, we simply model the pooling parameter as follows:

$$k_l = \max(\, k_{top},\, \lceil \frac{L-l}{L} s \rceil \,) \qquad (4)$$

where l is the number of the current convolutional layer to which the pooling is applied and L is the total number of convolutional layers in the network; k_{top} is the fixed pooling parameter for the topmost convolutional layer (Sect. 3.2). For instance, in a network with three convolutional layers and $k_{top} = 3$, for an input sentence of length $s = 18$, the pooling parameter at the first layer is $k_1 = 12$ and the pooling parameter at the second layer is $k_2 = 6$; the third layer has the fixed pooling parameter $k_3 = k_{top} = 3$. Equation 4 is a model of the number of values needed to describe the relevant parts of the progression of an l-th order feature over a sentence of length s. For an example in sentiment prediction, according to the equation a first order feature such as a positive word occurs *at most* k_1 times in a sentence of length s, whereas a second order feature such as a negated phrase or clause occurs at most k_2 times.

3.4 Non-linear Feature Function

After (dynamic) k-max pooling is applied to the result of a convolution, a bias $\mathbf{b} \in \mathbb{R}^d$ and a non-linear function g are applied component-wise to the pooled matrix. There is a single bias value for each row of the pooled matrix.

If we temporarily ignore the pooling layer, we may state how one computes each d-dimensional column a in the matrix $\mathbf{a}$ resulting after the convolutional and non-linear layers. Define $\mathbf{M}$ to be the matrix of diagonals:

$$\mathbf{M} = [\text{diag}(\mathbf{m}_{:,1}), \ldots, \text{diag}(\mathbf{m}_{:,m})] \qquad (5)$$

where $\mathbf{m}$ are the weights of the d filters of the wide convolution. Then after the first pair of a convolutional and a non-linear layer, each column a in the matrix $\mathbf{a}$ is obtained as follows, for some index j:

$$a = g\left(\mathbf{M} \begin{bmatrix} \mathbf{w}_j \\ \vdots \\ \mathbf{w}_{j+m-1} \end{bmatrix} + \mathbf{b}\right) \qquad (6)$$

Here a is a column of first order features. Second order features are similarly obtained by applying Eq. 6 to a sequence of first order features $a_j, ..., a_{j+m'-1}$ with another weight matrix $\mathbf{M}'$. Barring pooling, Eq. 6 represents a core aspect of the feature extraction function and has a rather general form that we return to below. Together with pooling, the feature function induces position invariance and makes the range of higher-order features variable.

3.5 Multiple Feature Maps

So far we have described how one applies a wide convolution, a (dynamic) k-max pooling layer and a non-linear function to the input sentence matrix to obtain a first order *feature map*. The three operations can be repeated to yield feature maps of increasing order and a network of increasing depth. We denote a feature map of the i-th order by $\mathbf{F}^i$. As in convolutional networks for object recognition, to increase the number of learnt feature detectors of a certain order, multiple feature maps $\mathbf{F}^i_1, \ldots, \mathbf{F}^i_n$ may be computed in parallel at the same layer. Each feature map $\mathbf{F}^i_j$ is computed by convolving a distinct set of filters arranged in a matrix $\mathbf{m}^i_{j,k}$ with each feature map $\mathbf{F}^{i-1}_k$ of the lower order $i-1$ and summing the results:

$$\mathbf{F}_j^i = \sum_{k=1}^{n} \mathbf{m}_{j,k}^i * \mathbf{F}_k^{i-1} \quad (7)$$

where $*$ indicates the wide convolution. The weights $\mathbf{m}_{j,k}^i$ form an order-4 tensor. After the wide convolution, first dynamic k-max pooling and then the non-linear function are applied individually to each map.

3.6 Folding

In the formulation of the network so far, feature detectors applied to an individual row of the sentence matrix $\mathbf{s}$ can have many orders and create complex dependencies across the same rows in multiple feature maps. Feature detectors in different rows, however, are independent of each other until the top fully connected layer. Full dependence between different rows could be achieved by making $\mathbf{M}$ in Eq. 5 a full matrix instead of a sparse matrix of diagonals. Here we explore a simpler method called *folding* that does not introduce any additional parameters. After a convolutional layer and before (dynamic) k-max pooling, one just sums every two rows in a feature map component-wise. For a map of d rows, folding returns a map of $d/2$ rows, thus halving the size of the representation. With a folding layer, a feature detector of the i-th order depends now on two rows of feature values in the lower maps of order $i-1$. This ends the description of the DCNN.

4 Properties of the Sentence Model

We describe some of the properties of the sentence model based on the DCNN. We describe the notion of the *feature graph* induced over a sentence by the succession of convolutional and pooling layers. We briefly relate the properties to those of other neural sentence models.

4.1 Word and n-Gram Order

One of the basic properties is sensitivity to the order of the words in the input sentence. For most applications and in order to learn fine-grained feature detectors, it is beneficial for a model to be able to discriminate whether a specific n-gram occurs in the input. Likewise, it is beneficial for a model to be able to tell the *relative* position of the most relevant n-grams. The network is designed to capture these two aspects. The filters $\mathbf{m}$ of the wide convolution in the first layer can learn to recognise specific n-grams that have size less or equal to the filter width m; as we see in the experiments, m in the first layer is often set to a relatively large value such as 10. The subsequence of n-grams extracted by the generalised pooling operation induces invariance to absolute positions, but maintains their order and relative positions.

As regards the other neural sentence models, the class of NBoW models is by definition insensitive to word order. A sentence model based on a recurrent neural network is sensitive to word order, but it has a bias towards the latest words that it takes as input (Mikolov et al., 2011). This gives the RNN excellent performance at language modelling, but it is suboptimal for remembering at once the n-grams further back in the input sentence. Similarly, a recursive neural network is sensitive to word order but has a bias towards the topmost nodes in the tree; shallower trees mitigate this effect to some extent (Socher et al., 2013a). As seen in Sect. 2.3, the Max-TDNN is sensitive to word order, but max pooling only picks out a single n-gram feature in each row of the sentence matrix.

4.2 Induced Feature Graph

Some sentence models use internal or external structure to compute the representation for the input sentence. In a DCNN, the convolution and pooling layers induce an internal feature graph over the input. A node from a layer is connected to a node from the next higher layer if the lower node is involved in the convolution that computes the value of the higher node. Nodes that are not selected by the pooling operation at a layer are dropped from the graph. After the last pooling layer, the remaining nodes connect to a single topmost root. The induced graph is a connected, directed acyclic graph with weighted edges and a root node; two equivalent representations of an induced graph are given in Fig. 1. In a DCNN without folding layers, each of the d rows of the sentence matrix induces a subgraph that joins the other subgraphs only at the root node. Each subgraph may have a different shape that reflects the kind of relations that are detected in that subgraph. The effect of folding layers is to join pairs of subgraphs at lower layers before the top root node.

Convolutional networks for object recognition also induce a feature graph over the input image. What makes the feature graph of a DCNN peculiar is the global range of the pooling operations. The (dynamic) k-max pooling operator can draw together features that correspond to words that are many positions apart in the sentence. Higher-order features have highly variable ranges that can be ei-

ther short and focused or global and long as the input sentence. Likewise, the edges of a subgraph in the induced graph reflect these varying ranges. The subgraphs can either be localised to one or more parts of the sentence or spread more widely across the sentence. This structure is internal to the network and is defined by the forward propagation of the input through the network.

Of the other sentence models, the NBoW is a shallow model and the RNN has a linear chain structure. The subgraphs induced in the Max-TDNN model have a single fixed-range feature obtained through max pooling. The recursive neural network follows the structure of an external parse tree. Features of variable range are computed at each node of the tree combining one or more of the children of the tree. Unlike in a DCNN, where one learns a clear hierarchy of feature orders, in a RecNN low order features like those of single words can be directly combined with higher order features computed from entire clauses. A DCNN generalises many of the structural aspects of a RecNN. The feature extraction function as stated in Eq. 6 has a more general form than that in a RecNN, where the value of m is generally 2. Likewise, the induced graph structure in a DCNN is more general than a parse tree in that it is not limited to syntactically dictated phrases; the graph structure can capture short or long-range semantic relations between words that do not necessarily correspond to the syntactic relations in a parse tree. The DCNN has internal input-dependent structure and does not rely on externally provided parse trees, which makes the DCNN directly applicable to hard-to-parse sentences such as tweets and to sentences from any language.

5 Experiments

We test the network on four different experiments. We begin by specifying aspects of the implementation and the training of the network. We then relate the results of the experiments and we inspect the learnt feature detectors.

5.1 Training

In each of the experiments, the top layer of the network has a fully connected layer followed by a softmax non-linearity that predicts the probability distribution over classes given the input sentence. The network is trained to minimise the cross-entropy of the predicted and true distributions; the objective includes an L_2 regularisation term over the parameters. The set of parameters comprises the word embeddings, the filter weights and the weights from the fully connected layers. The network is trained with mini-batches by back-propagation and the gradient-based optimisation is performed using the Adagrad update rule (Duchi et al., 2011). Using the well-known convolution theorem, we can compute fast one-dimensional linear convolutions at all rows of an input matrix by using Fast Fourier Transforms. To exploit the parallelism of the operations, we train the network on a GPU. A Matlab implementation processes multiple millions of input sentences per hour on one GPU, depending primarily on the number of layers used in the network.

Classifier	Fine-grained (%)	Binary (%)
NB	41.0	81.8
BINB	41.9	83.1
SVM	40.7	79.4
RECNTN	45.7	85.4
MAX-TDNN	37.4	77.1
NBOW	42.4	80.5
DCNN	**48.5**	**86.8**

Table 1: Accuracy of sentiment prediction in the movie reviews dataset. The first four results are reported from Socher et al. (2013b). The baselines NB and BINB are Naive Bayes classifiers with, respectively, unigram features and unigram and bigram features. SVM is a support vector machine with unigram and bigram features. RECNTN is a recursive neural network with a tensor-based feature function, which relies on external structural features given by a parse tree and performs best among the RecNNs.

5.2 Sentiment Prediction in Movie Reviews

The first two experiments concern the prediction of the sentiment of movie reviews in the Stanford Sentiment Treebank (Socher et al., 2013b). The output variable is binary in one experiment and can have five possible outcomes in the other: negative, somewhat negative, neutral, somewhat positive, positive. In the binary case, we use the given splits of 6920 training, 872 development and 1821 test sentences. Likewise, in the fine-grained case, we use the standard 8544/1101/2210 splits. Labelled phrases that occur as subparts of the training sentences are treated as independent training instances. The size of the vocabulary is 15448.

Table 1 details the results of the experiments.

Classifier	Features	Acc. (%)
HIER	unigram, POS, head chunks NE, semantic relations	91.0
MAXENT	unigram, bigram, trigram POS, chunks, NE, supertags CCG parser, WordNet	92.6
MAXENT	unigram, bigram, trigram POS, wh-word, head word word shape, parser hypernyms, WordNet	93.6
SVM	unigram, POS, wh-word head word, parser hypernyms, WordNet 60 hand-coded rules	95.0
MAX-TDNN	unsupervised vectors	84.4
NBOW	unsupervised vectors	88.2
DCNN	unsupervised vectors	93.0

Table 2: Accuracy of six-way question classification on the TREC questions dataset. The second column details the external features used in the various approaches. The first four results are respectively from Li and Roth (2002), Blunsom et al. (2006), Huang et al. (2008) and Silva et al. (2011).

In the three neural sentence models—the Max-TDNN, the NBoW and the DCNN—the word vectors are parameters of the models that are randomly initialised; their dimension d is set to 48. The Max-TDNN has a filter of width 6 in its narrow convolution at the first layer; shorter phrases are padded with zero vectors. The convolutional layer is followed by a non-linearity, a max-pooling layer and a softmax classification layer. The NBoW sums the word vectors and applies a non-linearity followed by a softmax classification layer. The adopted non-linearity is the $\tanh$ function. The hyper parameters of the DCNN are as follows. The binary result is based on a DCNN that has a wide convolutional layer followed by a folding layer, a dynamic k-max pooling layer and a non-linearity; it has a second wide convolutional layer followed by a folding layer, a k-max pooling layer and a non-linearity. The width of the convolutional filters is 7 and 5, respectively. The value of k for the top k-max pooling is 4. The number of feature maps at the first convolutional layer is 6; the number of maps at the second convolutional layer is 14. The network is topped by a softmax classification layer. The DCNN for the fine-grained result has the same architecture, but the filters have size 10 and 7, the top pooling parameter k is 5 and the number of maps is, respectively, 6 and 12. The networks use the $\tanh$ non-linear function. At training time we apply dropout to the penultimate layer after the last $\tanh$ non-linearity (Hinton et al., 2012).

Classifier	Accuracy (%)
SVM	81.6
BINB	82.7
MAXENT	83.0
MAX-TDNN	78.8
NBOW	80.9
DCNN	**87.4**

Table 3: Accuracy on the Twitter sentiment dataset. The three non-neural classifiers are based on unigram and bigram features; the results are reported from (Go et al., 2009).

We see that the DCNN significantly outperforms the other neural and non-neural models. The NBoW performs similarly to the non-neural n-gram based classifiers. The Max-TDNN performs worse than the NBoW likely due to the excessive pooling of the max pooling operation; the latter discards most of the sentiment features of the words in the input sentence. Besides the RecNN that uses an external parser to produce structural features for the model, the other models use n-gram based or neural features that do not require external resources or additional annotations. In the next experiment we compare the performance of the DCNN with those of methods that use heavily engineered resources.

5.3 Question Type Classification

As an aid to question answering, a question may be classified as belonging to one of many question types. The TREC questions dataset involves six different question types, e.g. whether the question is about a location, about a person or about some numeric information (Li and Roth, 2002). The training dataset consists of 5452 labelled questions whereas the test dataset consists of 500 questions.

The results are reported in Tab. 2. The non-neural approaches use a classifier over a large number of manually engineered features and hand-coded resources. For instance, Blunsom et al. (2006) present a Maximum Entropy model that relies on 26 sets of syntactic and semantic features including unigrams, bigrams, trigrams, POS tags, named entity tags, structural relations from a CCG parse and WordNet synsets. We evaluate the three neural models on this dataset with mostly the same hyper-parameters as in the binary senti-

POSITIVE						
lovely	comedic	moments	and	several	fine	performances
good	script	,	good	dialogue	,	funny
sustains	throughout	is	daring	,	inventive	and
well	written	,	nicely	acted	and	beautifully
remarkably	solid	and	subtly	satirical	tour	de
NEGATIVE						
,	nonexistent	plot	and	pretentious	visual	style
it	fails	the	most	basic	test	as
so	stupid	,	so	ill	conceived	,
,	too	dull	and	pretentious	to	be
hood	rats	butt	their	ugly	heads	in

'NOT'						
n't	have	any	huge	laughs	in	its
no	movement	,	no	,	not	much
n't	stop	me	from	enjoying	much	of
not	that	kung	pow	is	n't	funny
not	a	moment	that	is	not	false
'TOO'						
,	too	dull	and	pretentious	to	be
either	too	serious	or	too	lighthearted	,
too	slow	,	too	long	and	too
feels	too	formulaic	and	too	familiar	to
is	too	predictable	and	too	self	conscious

Figure 4: Top five 7-grams at four feature detectors in the first layer of the network.

ment experiment of Sect. 5.2. As the dataset is rather small, we use lower-dimensional word vectors with $d = 32$ that are initialised with embeddings trained in an unsupervised way to predict contexts of occurrence (Turian et al., 2010). The DCNN uses a single convolutional layer with filters of size 8 and 5 feature maps. The difference between the performance of the DCNN and that of the other high-performing methods in Tab. 2 is not significant ($p < 0.09$). Given that the only labelled information used to train the network is the training set itself, it is notable that the network matches the performance of state-of-the-art classifiers that rely on large amounts of engineered features and rules and hand-coded resources.

5.4 Twitter Sentiment Prediction with Distant Supervision

In our final experiment, we train the models on a large dataset of tweets, where a tweet is automatically labelled as positive or negative depending on the emoticon that occurs in it. The training set consists of 1.6 million tweets with emoticon-based labels and the test set of about 400 hand-annotated tweets. We preprocess the tweets minimally following the procedure described in Go et al. (2009); in addition, we also lowercase all the tokens. This results in a vocabulary of 76643 word types. The architecture of the DCNN and of the other neural models is the same as the one used in the binary experiment of Sect. 5.2. The randomly initialised word embeddings are increased in length to a dimension of $d = 60$. Table 3 reports the results of the experiments. We see a significant increase in the performance of the DCNN with respect to the non-neural n-gram based classifiers; in the presence of large amounts of training data these classifiers constitute particularly strong baselines. We see that the ability to train a sentiment classifier on automatically extracted emoticon-based labels extends to the DCNN and results in highly accurate performance. The difference in performance between the DCNN and the NBoW further suggests that the ability of the DCNN to both capture features based on long n-grams and to hierarchically combine these features is highly beneficial.

5.5 Visualising Feature Detectors

A filter in the DCNN is associated with a feature detector or neuron that learns during training to be particularly active when presented with a specific sequence of input words. In the first layer, the sequence is a continuous n-gram from the input sentence; in higher layers, sequences can be made of multiple separate n-grams. We visualise the feature detectors in the first layer of the network trained on the binary sentiment task (Sect. 5.2). Since the filters have width 7, for each of the 288 feature detectors we rank all 7-grams occurring in the validation and test sets according to their activation of the detector. Figure 5.2 presents the top five 7-grams for four feature detectors. Besides the expected detectors for positive and negative sentiment, we find detectors for particles such as 'not' that negate sentiment and such as 'too' that potentiate sentiment. We find detectors for multiple other notable constructs including 'all', 'or', 'with...that', 'as...as'. The feature detectors learn to recognise not just single n-grams, but patterns within n-grams that have syntactic, semantic or structural significance.

6 Conclusion

We have described a dynamic convolutional neural network that uses the dynamic k-max pooling operator as a non-linear subsampling function. The feature graph induced by the network is able to capture word relations of varying size. The network achieves high performance on question and sentiment classification without requiring external features as provided by parsers or other resources.

Acknowledgements

We thank Nando de Freitas and Yee Whye Teh for great discussions on the paper. This work was supported by a Xerox Foundation Award, EPSRC grant number EP/F042728/1, and EPSRC grant number EP/K036580/1.

References

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *EMNLP*, pages 1183–1193. ACL.

Phil Blunsom, Krystle Kocik, and James R. Curran. 2006. Question classification with log-linear models. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 615–616, New York, NY, USA. ACM.

Daoud Clarke. 2012. A context-theoretic framework for compositionality in distributional semantics. *Computational Linguistics*, 38(1):41–71.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical Foundations for a Compositional Distributional Model of Meaning. March.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *International Conference on Machine Learning, ICML*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July.

Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. *Proceedings of the Conference on Empirical Methods in Natural Language Processing - EMNLP '08*, (October):897.

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

Felix A. Gers and Jrgen Schmidhuber. 2001. Lstm recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks*, 12(6):1333–1340.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *Processing*, pages 1–6.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1394–1404. Association for Computational Linguistics.

Edward Grefenstette. 2013. Category-theoretic quantitative compositional distributional models of natural language semantics. *arXiv preprint arXiv:1311.1539*.

Emiliano Guevara. 2010. Modelling Adjective-Noun Compositionality by Regression. *ESSLLI'10 Workshop on Compositionality and Distributional Semantic Models*.

Karl Moritz Hermann and Phil Blunsom. 2013. The Role of Syntax in Vector Space Models of Compositional Semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Sofia, Bulgaria, August. Association for Computational Linguistics. Forthcoming.

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

Geoffrey E. Hinton. 1989. Connectionist learning procedures. *Artif. Intell.*, 40(1-3):185–234.

Zhiheng Huang, Marcus Thint, and Zengchang Qin. 2008. Question classification using head words and their hypernyms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 927–936, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

Nal Kalchbrenner and Phil Blunsom. 2013b. Recurrent Convolutional Neural Networks for Discourse Compositionality. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, Sofia, Bulgaria, August. Association for Computational Linguistics.

Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. 2013. Prior disambiguation of word tensors for constructing sentence vectors. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Seattle, USA, October.

Andreas Küchler and Christoph Goller. 1996. Inductive learning in symbolic domains using structure-driven recurrent neural networks. In Günther Görz and Steffen Hölldobler, editors, *KI*, volume 1137 of *Lecture Notes in Computer Science*, pages 183–197. Springer.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. In *SLT*, pages 234–239.

Tomas Mikolov, Stefan Kombrink, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *ICASSP*, pages 5528–5531. IEEE.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL*, volume 8.

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

Jordan B. Pollack. 1990. Recursive distributed representations. *Artificial Intelligence*, 46:77–105.

Holger Schwenk. 2012. Continuous space translation models for phrase-based statistical machine translation. In *COLING (Posters)*, pages 1071–1080.

Joo Silva, Lusa Coheur, AnaCristina Mendes, and Andreas Wichert. 2011. From symbolic to subsymbolic information in question classification. *Artificial Intelligence Review*, 35(2):137–154.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Richard Socher, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. 2013a. Grounded Compositional Semantics for Finding and Describing Images with Sentences. In *Transactions of the Association for Computational Linguistics (TACL)*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Stroudsburg, PA, October. Association for Computational Linguistics.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics.

Peter Turney. 2012. Domain and function: A dual-space model of semantic relations and compositions. *J. Artif. Intell. Res.(JAIR)*, 44:533–585.

Alexander Waibel, Toshiyuki Hanazawa, Geofrey Hinton, Kiyohiro Shikano, and Kevin J. Lang. 1990. Readings in speech recognition. chapter Phoneme Recognition Using Time-delay Neural Networks, pages 393–404. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Fabio Massimo Zanzotto, Ioannis Korkontzelos, Francesca Fallucchi, and Suresh Manandhar. 2010. Estimating linear models for compositional distributional semantics. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1263–1271. Association for Computational Linguistics.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*, pages 658–666. AUAI Press.

Online Learning in Tensor Space

Yuan Cao **Sanjeev Khudanpur**
Center for Language & Speech Processing and Human Language Technology Center of Excellence
The Johns Hopkins University
Baltimore, MD, USA, 21218
{yuan.cao, khudanpur}@jhu.edu

Abstract

We propose an online learning algorithm based on tensor-space models. A tensor-space model represents data in a compact way, and via rank-1 approximation the weight tensor can be made highly structured, resulting in a significantly smaller number of free parameters to be estimated than in comparable vector-space models. This regularizes the model complexity and makes the tensor model highly effective in situations where a large feature set is defined but very limited resources are available for training. We apply with the proposed algorithm to a parsing task, and show that even with very little training data the learning algorithm based on a tensor model performs well, and gives significantly better results than standard learning algorithms based on traditional vector-space models.

1 Introduction

Many NLP applications use models that try to incorporate a large number of linguistic features so that as much human knowledge of language can be brought to bear on the (prediction) task as possible. This also makes training the model parameters a challenging problem, since the amount of labeled training data is usually small compared to the size of feature sets: the feature weights cannot be estimated reliably.

Most traditional models are linear models, in the sense that both the features of the data and model parameters are represented as vectors in a vector space. Many learning algorithms applied to NLP problems, such as the Perceptron (Collins, 2002), MIRA (Crammer et al., 2006; McDonald et al., 2005; Chiang et al., 2008), PRO (Hopkins and May, 2011), RAMPION (Gimpel and Smith, 2012) etc., are based on vector-space models. Such models require learning individual feature weights directly, so that the number of parameters to be estimated is identical to the size of the feature set. When millions of features are used but the amount of labeled data is limited, it can be difficult to precisely estimate each feature weight.

In this paper, we shift the model from vector-space to tensor-space. Data can be represented in a compact and structured way using tensors as containers. Tensor representations have been applied to computer vision problems (Hazan et al., 2005; Shashua and Hazan, 2005) and information retrieval (Cai et al., 2006a) a long time ago. More recently, it has also been applied to parsing (Cohen and Collins, 2012; Cohen and Satta, 2013) and semantic analysis (Van de Cruys et al., 2013). A linear tensor model represents both features and weights in tensor-space, hence the weight tensor can be factorized and approximated by a linear sum of rank-1 tensors. This low-rank approximation imposes structural constraints on the feature weights and can be regarded as a form of regularization. With this representation, we no longer need to estimate individual feature weights directly but only a small number of "bases" instead. This property makes the the tensor model very effective when training a large number of feature weights in a low-resource environment. On the other hand, tensor models have many more degrees of "design freedom" than vector space models. While this makes them very flexible, it also creates much difficulty in designing an optimal tensor structure for a given training set.

We give detailed description of the tensor space

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 666–675,
Baltimore, Maryland, USA, June 23-25 2014.

model in Section 2. Several issues that come with the tensor model construction are addressed in Section 3. A tensor weight learning algorithm is then proposed in 4. Finally we give our experimental results on a parsing task and analysis in Section 5.

2 Tensor Space Representation

Most of the learning algorithms for NLP problems are based on vector space models, which represent data as vectors $\boldsymbol{\phi} \in \mathbb{R}^n$, and try to learn feature weight vectors $\boldsymbol{w} \in \mathbb{R}^n$ such that a linear model $y = \boldsymbol{w} \cdot \boldsymbol{\phi}$ is able to discriminate between, say, good and bad hypotheses. While this is a natural way of representing data, it is not the only choice. Below, we reformulate the model from vector to tensor space.

2.1 Tensor Space Model

A tensor is a multidimensional array, and is a generalization of commonly used algebraic objects such as vectors and matrices. Specifically, a vector is a 1^{st} order tensor, a matrix is a 2^{nd} order tensor, and data organized as a rectangular cuboid is a 3^{rd} order tensor etc. In general, a D^{th} order tensor is represented as $\mathcal{T} \in \mathbb{R}^{n_1 \times n_2 \times \ldots n_D}$, and an entry in $\mathcal{T}$ is denoted by $\mathcal{T}_{i_1,i_2,\ldots,i_D}$. Different dimensions of a tensor $1, 2, \ldots, D$ are named modes of the tensor.

Using a D^{th} order tensor as container, we can assign each feature of the task a D-dimensional index in the tensor and represent the data as tensors. Of course, shifting from a vector to a tensor representation entails several additional degrees of freedom, e.g., the order D of the tensor and the sizes $\{n_d\}_{d=1}^{D}$ of the modes, which must be addressed when selecting a tensor model. This will be done in Section 3.

2.2 Tensor Decomposition

Just as a matrix can be decomposed as a linear combination of several rank-1 matrices via SVD, tensors also admit decompositions[1] into linear combinations of "rank-1" tensors. A D^{th} order tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \ldots n_D}$ is rank-1 if it can be written as the outer product of D vectors, i.e.

$$\mathcal{A} = \mathbf{a}^1 \otimes \mathbf{a}^2 \otimes, \ldots, \otimes \mathbf{a}^D,$$

where $\mathbf{a}^i \in \mathbb{R}^{n_d}, 1 \leq d \leq D$. A D^{th} order tensor $\mathcal{T} \in \mathbb{R}^{n_1 \times n_2 \times \ldots n_D}$ can be factorized into a sum of component rank-1 tensors as

$$\mathcal{T} = \sum_{r=1}^{R} \mathcal{A}_r = \sum_{r=1}^{R} \mathbf{a}_r^1 \otimes \mathbf{a}_r^2 \otimes, \ldots, \otimes \mathbf{a}_r^D$$

where R, called the rank of the tensor, is the minimum number of rank-1 tensors whose sum equals $\mathcal{T}$. Via decomposition, one may approximate a tensor by the sum of H major rank-1 tensors with $H \leq R$.

2.3 Linear Tensor Model

In tensor space, a linear model may be written (ignoring a bias term) as

$$f(\boldsymbol{W}) = \boldsymbol{W} \circ \boldsymbol{\Phi},$$

where $\boldsymbol{\Phi} \in \mathbb{R}^{n_1 \times n_2 \times \ldots n_D}$ is the feature tensor, $\boldsymbol{W}$ is the corresponding weight tensor, and $\circ$ denotes the Hadamard product. If $\boldsymbol{W}$ is further decomposed as the sum of H major component rank-1 tensors, i.e. $\boldsymbol{W} \approx \sum_{h=1}^{H} \boldsymbol{w}_h^1 \otimes \boldsymbol{w}_h^2 \otimes, \ldots, \otimes \boldsymbol{w}_h^D$, then

$$\begin{aligned} & f(\boldsymbol{w}_1^1, \ldots, \boldsymbol{w}_1^D, \ldots, \boldsymbol{w}_h^1, \ldots, \boldsymbol{w}_h^D) \\ &= \sum_{h=1}^{H} \boldsymbol{\Phi} \times_1 \boldsymbol{w}_h^1 \times_2 \boldsymbol{w}_h^2 \ldots \times_D \boldsymbol{w}_h^D, \quad (1) \end{aligned}$$

where $\times_l$ is the l-mode product operator between a D^{th} order tensor $\mathcal{T}$ and a vector $\mathbf{a}$ of dimension n_d, yielding a $(D-1)^{\text{th}}$ order tensor such that

$$\begin{aligned} & (\mathcal{T} \times_l \mathbf{a})_{i_1,\ldots,i_{l-1},i_{l+1},\ldots,i_D} \\ &= \sum_{i_l=1}^{n_d} \mathcal{T}_{i_1,\ldots,i_{l-1},i_l,i_{l+1},\ldots,i_D} \cdot a_{i_l}. \end{aligned}$$

The linear tensor model is illustrated in Figure 1.

2.4 Why Learning in Tensor Space?

So what is the advantage of learning with a tensor model instead of a vector model? Consider the case where we have defined 1,000,000 features for our task. A vector space linear model requires estimating 1,000,000 free parameters. However if we use a 2^{nd} order tensor model, organize the features into a 1000×1000 matrix $\boldsymbol{\Phi}$, and use just

[1] The form of tensor decomposition defined here is named as CANDECOMP/PARAFAC(CP) decomposition (Kolda and Bader, 2009). Another popular form of tensor decomposition is called Tucker decomposition, which decomposes a tensor into a core tensor multiplied by a matrix along each mode. We focus only on the CP decomposition in this paper.

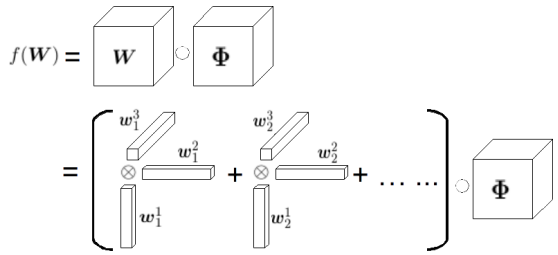


Figure 1: A 3^{rd} order linear tensor model. The feature weight tensor $\boldsymbol{W}$ can be decomposed as the sum of a sequence of rank-1 component tensors.

one rank-1 matrix to approximate the weight tensor, then the linear model becomes

$$f(\boldsymbol{w}_1, \boldsymbol{w}_2) = \boldsymbol{w}_1^T \boldsymbol{\Phi} \boldsymbol{w}_2,$$

where $\boldsymbol{w}_1, \boldsymbol{w}_2 \in \mathbb{R}^{1000}$. That is to say, now we only need to estimate 2000 parameters!

In general, if V features are defined for a learning problem, and we (i) organize the feature set as a tensor $\boldsymbol{\Phi} \in \mathbb{R}^{n_1 \times n_2 \times \ldots n_D}$ and (ii) use H component rank-1 tensors to approximate the corresponding target weight tensor. Then the total number of parameters to be learned for this tensor model is $H \sum_{d=1}^{D} n_d$, which is usually much smaller than $V = \prod_{d=1}^{D} n_d$ for a traditional vector space model. Therefore we expect the tensor model to be more effective in a low-resource training environment.

Specifically, a vector space model assumes each feature weight to be a "free" parameter, and estimating them reliably could therefore be hard when training data are not sufficient or the feature set is huge. By contrast, a linear tensor model only needs to learn $H \sum_{d=1}^{D} n_d$ "bases" of the m feature weights instead of individual weights directly. The weight corresponding to the feature $\Phi_{i_1, i_2, \ldots, i_D}$ in the tensor model is expressed as

$$w_{i_1, i_2, \ldots, i_D} = \sum_{h=1}^{H} w_{h,i_1}^1 w_{h,i_2}^2 \ldots w_{h,i_D}^D, \quad (2)$$

where w_{h,i_j}^j is the i_j^{th} element in the vector $\boldsymbol{w}_h^j$.

In other words, a true feature weight is now approximated by a set of bases. This reminds us of the well-known low-rank matrix approximation of images via SVD, and we are applying similar techniques to approximate target feature weights, which is made possible only after we shift from vector to tensor space models.

This approximation can be treated as a form of model regularization, since the weight tensor is represented in a constrained form and made highly structured via the rank-1 tensor approximation. Of course, as we reduce the model complexity, e.g. by choosing a smaller and smaller H, the model's expressive ability is weakened at the same time. We will elaborate on this point in Section 3.1.

3 Tensor Model Construction

To apply a tensor model, we first need to convert the feature vector into a tensor $\boldsymbol{\Phi}$. Once the structure of $\boldsymbol{\Phi}$ is determined, the structure of $\boldsymbol{W}$ is fixed as well. As mentioned in Section 2.1, a tensor model has many more degrees of "design freedom" than a vector model, which makes the problem of finding a good tensor structure a non-trivial one.

3.1 Tensor Order

The order of a tensor affects the model in two ways: the expressiveness of the model and the number of parameters to be estimated. We assume $H = 1$ in the analysis below, noting that one can always add as many rank-1 component tensors as needed to approximate a tensor with arbitrary precision.

Obviously, the 1^{st} order tensor (vector) model is the most expressive, since it is structureless and any arbitrary set of numbers can always be represented exactly as a vector. The 2^{nd} order rank-1 tensor (rank-1 matrix) is less expressive because not every set of numbers can be organized into a rank-1 matrix. In general, a D^{th} order rank-1 tensor is more expressive than a $(D+1)^{\text{th}}$ order rank-1 tensor, as a lower-order tensor imposes less structural constraints on the set of numbers it can express. We formally state this fact as follows:

Theorem 1. *A set of real numbers that can be represented by a $(D+1)^{\text{th}}$ order tensor $\mathcal{Q}$ can also be represented by a D^{th} order tensor $\mathcal{P}$, provided $\mathcal{P}$ and $\mathcal{Q}$ have the same volume. But the reverse is not true.*

Proof. See appendix. □

On the other hand, tensor order also affects the number of parameters to be trained. Assuming that a D^{th} order has equal size on each mode (we will elaborate on this point in Section 3.2) and the volume (number of entries) of the tensor is fixed as V, then the total number of parameters

of the model is $DV^{\frac{1}{D}}$. This is a convex function of D, and the minimum[2] is reached at either $D^* = \lfloor \ln V \rfloor$ or $D^* = \lceil \ln V \rceil$.

Therefore, as D increases from 1 to D^*, we lose more and more of the expressive power of the model but reduce the number of parameters to be trained. However it would be a bad idea to choose a D beyond D^*. The optimal tensor order depends on the nature of the actual problem, and we tune this hyper-parameter on a held-out set.

3.2 Mode Size

The size n_d of each tensor mode, $d = 1, \ldots, D$, determines the structure of feature weights a tensor model can precisely represent, as well as the number of parameters to estimate (we also assume $H = 1$ in the analysis below). For example, if the tensor order is 2 and the volume V is 12, then we can either choose $n_1 = 3, n_2 = 4$ or $n_1 = 2, n_2 = 6$. For $n_1 = 3, n_2 = 4$, the numbers that can be precisely represented are divided into 3 groups, each having 4 numbers, that are scaled versions of one another. Similarly for $n_1 = 2, n_2 = 6$, the numbers can be divided into 2 groups with different scales. Obviously, the two possible choices of (n_1, n_2) also lead to different numbers of free parameters (7 vs. 8).

Given D and V, there are many possible combinations of $n_d, d = 1, \ldots, D$, and the optimal combination should indeed be determined by the structure of target features weights. However it is hard to know the structure of target feature weights before learning, and it would be impractical to try every possible combination of mode sizes, therefore we choose the criterion of determining the mode sizes as minimization of the total number of parameters, namely we solve the problem:

$$\min_{n_1,\ldots,n_D} \sum_{d=1}^{D} n_d \quad s.t \quad \prod_{d=1}^{D} n_d = V$$

The optimal solution is reached when $n_1 = n_2 = \ldots = n_D = V^{\frac{1}{D}}$. Of course it is not guaranteed that $V^{\frac{1}{D}}$ is an integer, therefore we choose $n_d = \lfloor V^{\frac{1}{D}} \rfloor$ or $\lceil V^{\frac{1}{D}} \rceil, d = 1, \ldots, D$ such that $\prod_{d=1}^{D} n_d \geq V$ and $\left[\prod_{d=1}^{D} n_d\right] - V$ is minimized. The $\left[\prod_{d=1}^{D} n_d\right] - V$ extra entries of the tensor correspond to no features and are used just for padding. Since for each n_d there are only two possible values to choose, we can simply enumerate all the possible 2^D (which is usually a small number) combinations of values and pick the one that matches the conditions given above. This way $n_1, \ldots, n_D$ are fully determined.

Here we are only following the principle of minimizing the parameter number. While this strategy might work well with small amount of training data, it is not guaranteed to be the best strategy in all cases, especially when more data is available we might want to increase the number of parameters, making the model more complex so that the data can be more precisely modeled. Ideally the mode size needs to be adaptive to the amount of training data as well as the property of target weights. A theoretically guaranteed optimal approach to determining the mode sizes remains an open problem, and will be explored in our future work.

3.3 Number of Rank-1 Tensors

The impact of using $H > 1$ rank-1 tensors is obvious: a larger H increases the model complexity and makes the model more expressive, since we are able to approximate target weight tensor with smaller error. As a trade-off, the number of parameters and training complexity will be increased. To find out the optimal value of H for a given problem, we tune this hyper-parameter too on a held-out set.

3.4 Vector to Tensor Mapping

Finally, we need to find a way to map the original feature vector to a tensor, i.e. to associate each feature with an index in the tensor. Assuming the tensor volume V is the same as the number of features, then there are in all $V!$ ways of mapping, which is an intractable number of possibilities even for modest sized feature sets, making it impractical to carry out a brute force search. However while we are doing the mapping, we hope to arrange the features in a way such that the corresponding target weight tensor has approximately a low-rank structure, this way it can be well approximated by very few component rank-1 tensors.

Unfortunately we have no knowledge about the target weights in advance, since that is what we need to learn after all. As a way out, we first run a simple vector-model based learning algorithm (say the Perceptron) on the training data and estimate a weight vector, which serves as a "surro-

[2]The optimal integer solution can be determined simply by comparing the two function values.

gate" weight vector. We then use this surrogate vector to guide the design of the mapping. Ideally we hope to find a permutation of the surrogate weights to map to a tensor in such a way that the tensor has a rank as low as possible. However matrix rank minimization is in general a hard problem (Fazel, 2002). Therefore, we follow an approximate algorithm given in Figure 2a, whose main idea is illustrated via an example in Figure 2b.

Basically, what the algorithm does is to divide the surrogate weights into hierarchical groups such that groups on the same level are approximately proportional to each other. Using these groups as units we are able to "fill" the tensor in a hierarchical way. The resulting tensor will have an approximate low-rank structure, provided that the sorted feature weights have roughly group-wise proportional relations.

For comparison, we also experimented a trivial solution which maps each entry of the feature tensor to the tensor just in sequential order, namely ϕ_0 is mapped to $\Phi_{0,0,\ldots,0}$, ϕ_1 is mapped to $\Phi_{0,0,\ldots,1}$ etc. This of course ignores correlation between features since the original feature order in the vector could be totally meaningless, and this strategy is not expected to be a good solution for vector to tensor mapping.

4 Online Learning Algorithm

We now turn to the problem of learning the feature weight tensor. Here we propose an online learning algorithm similar to MIRA but modified to accommodate tensor models.

Let the model be $f(\boldsymbol{T}) = \boldsymbol{T} \circ \boldsymbol{\Phi}(x, y)$, where $\boldsymbol{T} = \sum_{h=1}^{H} \boldsymbol{w}_h^1 \otimes \boldsymbol{w}_h^2 \otimes, \ldots, \otimes \boldsymbol{w}_h^D$ is the weight tensor, $\boldsymbol{\Phi}(x, y)$ is the feature tensor for an input-output pair (x, y). Training samples $(x_i, y_i), i = 1, \ldots, m$, where x_i is the input and y_i is the reference or oracle hypothesis, are fed to the weight learning algorithm in sequential order. A prediction z_t is made by the model T_t at time t from a set of candidates $\mathcal{Z}(x_t)$, and the model updates the weight tensor by solving the following problem:

$$\min_{\boldsymbol{T} \in \mathbb{R}^{n_1 \times n_2 \times \ldots n_D}} \frac{1}{2}\|\boldsymbol{T} - \boldsymbol{T}_t\|^2 + C\xi \quad (3)$$
$$s.t.$$
$$\mathscr{L}_t \leq \xi, \xi \geq 0$$

where $\boldsymbol{T}$ is a decomposed weight tensor and

$$\mathscr{L}_t = \boldsymbol{T} \circ \boldsymbol{\Phi}(x_t, z_t) - \boldsymbol{T} \circ \boldsymbol{\Phi}(x_t, y_t) + \rho(y_t, z_t)$$

Input:

Tensor order D, tensor volume V, mode size $n_d, d = 1, \ldots, D$, surrogate weight vector $\boldsymbol{v}$

Let

$\boldsymbol{v}^+ = [v_1^+, \ldots, v_p^+]$ be the non-negative part of $\boldsymbol{v}$

$\boldsymbol{v}^- = [v_1^-, \ldots, v_q^-]$ be the negative part of $\boldsymbol{v}$

Algorithm:

$\tilde{\boldsymbol{v}}^+$ = *sort*($\boldsymbol{v}^+$) in descending order

$\tilde{\boldsymbol{v}}^-$ = *sort*($\boldsymbol{v}^-$) in ascending order

$u = V/n_D$

$e = p - mod(p, u)$, $f = q - mod(q, u)$

Construct vector

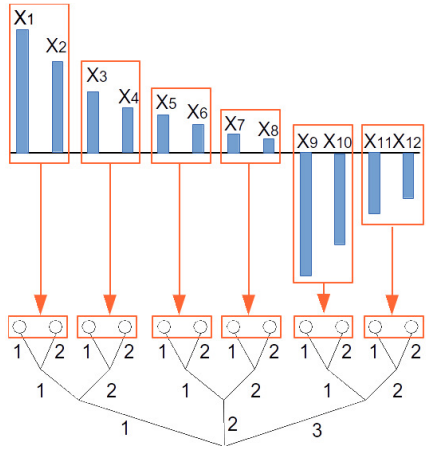
$\mathbf{X} = [\tilde{v}_1^+, \ldots, \tilde{v}_e^+, \tilde{v}_1^-, \ldots, \tilde{v}_f^-,$
$\tilde{v}_{e+1}^+, \ldots, \tilde{v}_p^+, \tilde{v}_{f+1}^-, \ldots, \tilde{v}_q^-]$

Map $X_a, a = 1, \ldots, p + q$ to the tensor entry $\mathcal{T}_{i_1,\ldots,i_D}$, such that

$$a = \sum_{d=1}^{D} (i_d - 1) l_{d-1} + 1$$

where $l_d = l_{d-1} n_d$, and $l_0 = 1$

(a) Mapping a surrogate weight vector to a tensor



(b) Illustration of the algorithm

Figure 2: Algorithm for mapping a surrogate weight vector X to a tensor. (2a) provides the algorithm; (2b) illustrates it by mapping a vector of length $V = 12$ to a $(n_1, n_2, n_3) = (2, 2, 3)$ tensor. The bars X_i represent the surrogate weights — after separately sorting the positive and negative parts — and the labels along a path of the tree correspond to the tensor-index of the weight represented by the leaf resulting from the mapping.

is the structured hinge loss.

This problem setting follows the same "passive-aggressive" strategy as in the original MIRA. To optimize the vectors $\boldsymbol{w}_h^d, h = 1, \ldots, H, d = 1, \ldots, D$, we use a similar iterative strategy as proposed in (Cai et al., 2006b). Basically, the idea is that instead of optimizing $\boldsymbol{w}_h^d$ all together, we optimize $\boldsymbol{w}_1^1, \boldsymbol{w}_1^2, \ldots, \boldsymbol{w}_H^D$ in turn. While we are updating one vector, the rest are fixed. For the problem setting given above, each of the sub-problems that need to be solved is convex, and according to (Cai et al., 2006b) the objective function value will decrease after each individual weight update and eventually this procedure will converge.

We now give this procedure in more detail. Denote the weight vector of the d^{th} mode of the h^{th} tensor at time t as $\boldsymbol{w}_{h,t}^d$. We will update the vectors in turn in the following order: $\boldsymbol{w}_{1,t}^1, \ldots, \boldsymbol{w}_{1,t}^D, \boldsymbol{w}_{2,t}^1, \ldots, \boldsymbol{w}_{2,t}^D, \ldots, \boldsymbol{w}_{H,t}^1, \ldots, \boldsymbol{w}_{H,t}^D$. Once a vector has been updated, it is fixed for future updates.

By way of notation, define

$$\begin{aligned}
&\boldsymbol{W}_{h,t}^d \\
&= \boldsymbol{w}_{h,t+1}^1 \otimes, \ldots, \otimes \boldsymbol{w}_{h,t+1}^{d-1} \otimes \boldsymbol{w}_{h,t}^d \otimes, \ldots, \otimes \boldsymbol{w}_{h,t}^D \\
&\quad \text{(and let } \boldsymbol{W}_{h,t}^{D+1} \triangleq \boldsymbol{w}_{h,t+1}^1 \otimes, \ldots, \otimes \boldsymbol{w}_{h,t+1}^D\text{)}, \\
&\widehat{\boldsymbol{W}}_{h,t}^d \\
&= \boldsymbol{w}_{h,t+1}^1 \otimes, \ldots, \otimes \boldsymbol{w}_{h,t+1}^{d-1} \otimes \boldsymbol{w}^d \otimes, \ldots, \otimes \boldsymbol{w}_{h,t}^D \\
&\quad \text{(where } \boldsymbol{w}^d \in \mathbb{R}^{n_d}\text{)},
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{T}_{h,t}^d &= \sum_{h'=1}^{h-1} \boldsymbol{W}_{h',t}^{D+1} + \boldsymbol{W}_{h,t}^d + \sum_{h'=h+1}^{H} \boldsymbol{W}_{h',t}^1 \quad (4) \\
\widehat{\boldsymbol{T}}_{h,t}^d &= \sum_{h'=1}^{h-1} \boldsymbol{W}_{h',t}^{D+1} + \widehat{\boldsymbol{W}}_{h,t}^d + \sum_{h'=h+1}^{H} \boldsymbol{W}_{h',t}^1
\end{aligned}$$

$$\begin{aligned}
&\boldsymbol{\phi}_{h,t}^d(x, y) \\
&= \boldsymbol{\Phi}(x, y) \times_2 \boldsymbol{w}_{h,t+1}^2 \ldots \times_{d-1} \boldsymbol{w}_{h,t+1}^{d-1} \times_{d+1} \\
&\quad \boldsymbol{w}_{h,t}^{d+1} \ldots \times_D \boldsymbol{w}_{h,t}^D \qquad (5)
\end{aligned}$$

In order to update from $\boldsymbol{w}_{h,t}^d$ to get $\boldsymbol{w}_{h,t+1}^d$, the sub-problem to solve is:

$$\begin{aligned}
&\min_{\boldsymbol{w}^d \in \mathbb{R}^{n_d}} \frac{1}{2} \|\widehat{\boldsymbol{T}}_{h,t}^d - \boldsymbol{T}_{h,t}^d\|^2 + C\xi \\
&= \min_{\boldsymbol{w}^d \in \mathbb{R}^{n_d}} \frac{1}{2} \|\widehat{\boldsymbol{W}}_{h,t}^d - \boldsymbol{W}_{h,t}^d\|^2 + C\xi \\
&= \min_{\boldsymbol{w}^d \in \mathbb{R}^{n_d}} \frac{1}{2} \beta_{h,t+1}^1 \ldots \beta_{h,t+1}^{d-1} \beta_{h,t}^{d+1} \ldots \beta_{h,t}^D \\
&\qquad \|\boldsymbol{w}^d - \boldsymbol{w}_{h,t}^d\|^2 + C\xi \\
&s.t. \qquad \mathscr{L}_{h,t}^d \le \xi, \xi \ge 0.
\end{aligned}$$

where

$$\begin{aligned}
\beta_{h,t}^d &= \|\boldsymbol{w}_{h,t}^d\|^2 \\
\mathscr{L}_{h,t}^d &= \widehat{\boldsymbol{T}}_{h,t}^d \circ \boldsymbol{\Phi}(x_t, z_t) - \widehat{\boldsymbol{T}}_{h,t}^d \circ \boldsymbol{\Phi}(x_t, y_t) \\
&\quad + \rho(y_t, z_t) \\
&= \boldsymbol{w}^d \cdot \left(\boldsymbol{\phi}_{h,t}^d(x_t, z_t) - \boldsymbol{\phi}_{h,t}^d(x_t, y_t) \right) \\
&\quad - \left(\sum_{h'=1}^{h-1} \boldsymbol{W}_{h',t}^{D+1} + \sum_{h'=h+1}^{H} \boldsymbol{W}_{h',t}^1 \right) \circ \\
&\quad (\boldsymbol{\Phi}(x_t, y_t) - \boldsymbol{\Phi}(x_t, z_t)) \\
&\quad + \rho(y_t, z_t)
\end{aligned}$$

Letting

$$\Delta \boldsymbol{\phi}_{h,t}^d \triangleq \boldsymbol{\phi}_{h,t}^d(x_t, y_t) - \boldsymbol{\phi}_{h,t}^d(x_t, z_t)$$

and

$$\begin{aligned}
s_{h,t}^d \triangleq &\left(\sum_{h'=1}^{h-1} \boldsymbol{W}_{h',t}^{D+1} + \sum_{h'=h+1}^{H} \boldsymbol{W}_{h',t}^1 \right) \circ \\
&(\boldsymbol{\Phi}(x_t, y_t) - \boldsymbol{\Phi}(x_t, z_t))
\end{aligned}$$

we may compactly write

$$\mathscr{L}_{h,t}^d = \rho(y_t, z_t) - s_{h,t}^d - \boldsymbol{w}^d \cdot \Delta \boldsymbol{\phi}_{h,t}^d.$$

This convex optimization problem is just like the original MIRA and may be solved in a similar way. The updating strategy for $\boldsymbol{w}_{h,t}^d$ is derived as

$$\begin{aligned}
&\boldsymbol{w}_{h,t+1}^d = \boldsymbol{w}_{h,t}^d + \tau \Delta \boldsymbol{\phi}_{h,t}^d \\
&\tau = \qquad (6) \\
&\min \left\{ C, \frac{\rho(y_t, z_t) - \boldsymbol{T}_{h,t}^d \circ (\boldsymbol{\Phi}(x_t, y_t) - \boldsymbol{\Phi}(x_t, z_t))}{\|\Delta \boldsymbol{\phi}_{h,t}^d\|^2} \right\}
\end{aligned}$$

The initial vectors $\boldsymbol{w}_{h,1}^i$ cannot be made all zero, since otherwise the l-mode product in Equation (5) would yield all zero $\boldsymbol{\phi}_{h,t}^d(x, y)$ and the model would never get a chance to be updated. Therefore, we initialize the entries of $\boldsymbol{w}_{h,1}^i$ uniformly such that the Frobenius-norm of the weight tensor $\boldsymbol{W}$ is unity.

We call the algorithm above "Tensor-MIRA" and abbreviate it as T-MIRA.

5 Experiments

In this section we shows empirical results of the training algorithm on a parsing task. We used the Charniak parser (Charniak et al., 2005) for our experiment, and we used the proposed algorithm to train the reranking feature weights. For comparison, we also investigated training the reranker with Perceptron and MIRA.

5.1 Experimental Settings

To simulate a low-resource training environment, our training sets were selected from sections 2-9 of the Penn WSJ treebank, section 24 was used as the held-out set and section 23 as the evaluation set. We applied the default settings of the parser. There are around $V = 1.33$ million features in all defined for reranking, and the n-best size for reranking is set to 50. We selected the parse with the highest f-score from the 50-best list as the oracle.

We would like to observe from the experiments how the amount of training data as well as different settings of the tensor degrees of freedom affects the algorithm performance. Therefore we tried all combinations of the following experimental parameters:

Parameters	Settings
Training data (m)	Sec. 2, 2-3, 2-5, 2-9
Tensor order (D)	2, 3, 4
# rank-1 tensors (H)	1, 2, 3
Vec. to tensor mapping	approximate, sequential

Here "approximate" and "sequential" means using, respectively, the algorithm given in Figure 2 and the sequential mapping mentioned in Section 3.4. According to the strategy given in 3.2, once the tensor order and number of features are fixed, the sizes of modes and total number of parameters to estimate are fixed as well, as shown in the tables below:

D	Size of modes	Number of parameters
2	1155×1155	2310
3	$110 \times 110 \times 111$	331
4	$34 \times 34 \times 34 \times 34$	136

5.2 Results and Analysis

The f-scores of the held-out and evaluation set given by T-MIRA as well as the Perceptron and MIRA baseline are given in Table 1. From the results, we have the following observations:

1. When very few labeled data are available for training (compared with the number of features), T-MIRA performs much better than the vector-based models MIRA and Perceptron. However as the amount of training data increases, the advantage of T-MIRA fades away, and vector-based models catch up. This is because the weight tensors learned by T-MIRA are highly structured, which significantly reduces model/training complexity and makes the learning process very effective in a low-resource environment, but as the amount of data increases, the more complex and expressive vector-based models adapt to the data better, whereas further improvements from the tensor model is impeded by its structural constraints, making it insensitive to the increase of training data.

2. To further contrast the behavior of T-MIRA, MIRA and Perceptron, we plot the f-scores on both the training and held-out sets given by these algorithms after each training epoch in Figure 3. The plots are for the experimental setting with mapping=surrogate, # rank-1 tensors=2, tensor order=2, training data=sections 2-3. It is clearly seen that both MIRA and Perceptron do much better than T-MIRA on the training set. Nevertheless, with a huge number of parameters to fit a limited amount of data, they tend to over-fit and give much worse results on the held-out set than T-MIRA does.

   As an aside, observe that MIRA consistently outperformed Perceptron, as expected.

3. Properties of linear tensor model: The heuristic vector-to-tensor mapping strategy given by Figure 2 gives consistently better results than the sequential mapping strategy, as expected.

   To make further comparison of the two strategies, in Figure 4 we plot the 20 largest singular values of the matrices which the surrogate weights (given by the Perceptron after running for 1 epoch) are mapped to by both strategies (from the experiment with training data sections 2-5). From the contrast between the largest and the 2^{nd}-largest singular values, it can be seen that the matrix generated

by the first strategy approximates a low-rank structure much better than the second strategy. Therefore, the performance of T-MIRA is influenced significantly by the way features are mapped to the tensor. If the corresponding target weight tensor has internal structure that makes it approximately low-rank, the learning procedure becomes more effective.

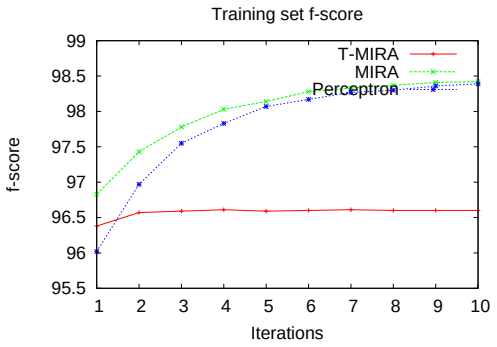
The best results are consistently given by 2^{nd} order tensor models, and the differences between the 3^{rd} and 4^{th} order tensors are not significant. As discussed in Section 3.1, although 3^{rd} and 4^{th} order tensors have less parameters, the benefit of reduced training complexity does not compensate for the loss of expressiveness. A 2^{nd} order tensor has already reduced the number of parameters from the original 1.33 million to only 2310, and it does not help to further reduce the number of parameters using higher order tensors.

4. As the amount of training data increases, there is a trend that the best results come from models with more rank-1 component tensors. Adding more rank-1 tensors increases the model's complexity and ability of expression, making the model more adaptive to larger data sets.

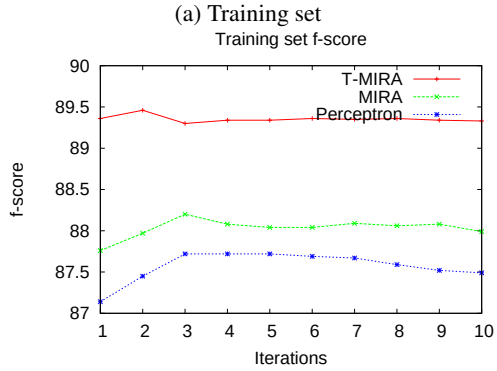
6 Conclusion and Future Work

In this paper, we reformulated the traditional linear vector-space models as tensor-space models, and proposed an online learning algorithm named Tensor-MIRA. A tensor-space model is a compact representation of data, and via rank-1 tensor approximation, the weight tensor can be made highly structured hence the number of parameters to be trained is significantly reduced. This can be regarded as a form of model regularization.Therefore, compared with the traditional vector-space models, learning in the tensor space is very effective when a large feature set is defined, but only small amount of training data is available. Our experimental results corroborated this argument.

As mentioned in Section 3.2, one interesting problem that merits further investigation is how to determine optimal mode sizes. The challenge of applying a tensor model comes from finding a proper tensor structure for a given problem, and



(a) Training set



(b) Held-out set

Figure 3: f-scores given by three algorithms on training and held-out set (see text for the setting).

the key to solving this problem is to find a balance between the model complexity (indicated by the order and sizes of modes) and the number of parameters. Developing a theoretically guaranteed approach of finding the optimal structure for a given task will make the tensor model not only perform well in low-resource environments, but adaptive to larger data sets.

7 Acknowledgements

This work was partially supported by IBM via DARPA/BOLT contract number HR0011-12-C-0015 and by the National Science Foundation via award number IIS-0963898.

References

Deng Cai , Xiaofei He , and Jiawei Han. 2006. Tensor Space Model for Document Analysis *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval(SIGIR)*, 625–626.

Deng Cai, Xiaofei He, and Jiawei Han. 2006. Learning with Tensor Representation *Technical Report, Department of Computer Science, University of Illinois at Urbana-Champaign.*

Mapping	Approximate									Sequential								
Rank-1 tensors	1			2			3			1			2			3		
Tensor order	2	3	4	2	3	4	2	3	4	2	3	4	2	3	4	2	3	4
Held-out score	**89.43**	89.16	89.22	89.16	89.21	89.24	89.27	89.14	89.24	**89.21**	88.90	88.89	89.13	88.88	88.88	89.15	88.87	88.99
Evaluation score	**89.83**									89.69								
MIRA	88.57																	
Percep	88.23																	

(a) Training data: Section 2 only

Mapping	Approximate									Sequential								
Rank-1 tensors	1			2			3			1			2			3		
Tensor order	2	3	4	2	3	4	2	3	4	2	3	4	2	3	4	2	3	4
Held-out score	89.26	89.06	89.12	**89.33**	89.11	89.19	89.18	89.14	89.15	89.2	89.01	88.82	**89.24**	88.94	88.95	89.19	88.91	88.98
Evaluation score	**90.02**									89.82								
MIRA	89.00																	
Percep	88.59																	

(b) Training data: Section 2-3

Mapping	Approximate									Sequential								
Rank-1 tensors	1			2			3			1			2			3		
Tensor order	2	3	4	2	3	4	2	3	4	2	3	4	2	3	4	2	3	4
Held-out score	89.40	89.44	89.17	**89.5**	89.37	89.18	89.47	89.32	89.18	89.23	89.03	88.93	**89.24**	88.98	88.94	89.16	89.01	88.85
Evaluation score	**89.96**									89.78								
MIRA	89.49																	
Percep	89.10																	

(c) Training data: Section 2-5

Mapping	Approximate									Sequential								
Rank-1 tensors	2			3			4			2			3			4		
Tensor order	2	3	4	2	3	4	2	3	4	2	3	4	2	3	4	2	3	4
Held-out score	89.43	89.23	89.06	89.37	89.23	89.1	**89.44**	89.22	89.06	89.21	88.92	88.94	89.23	88.94	88.93	**89.23**	88.95	88.93
Evaluation score	**89.95**									89.84								
MIRA	89.95																	
Percep	89.77																	

(d) Training data: Section 2-9

Table 1: Parsing f-scores. Tables (a) to (d) correspond to training data with increasing size. The upper-part of each table shows the T-MIRA results with different settings, the lower-part shows the MIRA and Perceptron baselines. The evaluation scores come from the settings indicated by the best held-out scores. The best results on the held-out and evaluation data are marked in bold.

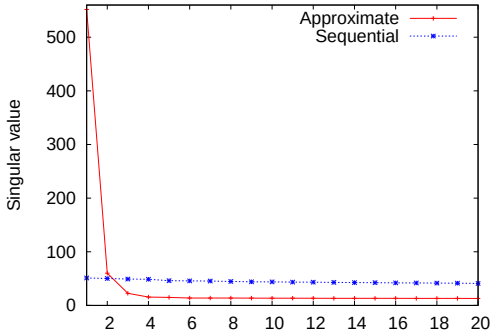


Figure 4: The top 20 singular values of the surrogate weight matrices given by two mapping algorithms.

Eugene Charniak, and Mark Johnson 2005. Coarse-to-fine n-Best Parsing and MaxEnt Discriminative Reranking *Proceedings of the 43th Annual Meeting on Association for Computational Linguistics(ACL)* 173–180.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online Large-Margin Training of Syntactic and Structural Translation Features *Proceedings of Empirical Methods in Natural Language Processing(EMNLP)*, 224–233.

Shay Cohen and Michael Collins. 2012. Tensor Decomposition for Fast Parsing with Latent-Variable PCFGs *Proceedings of Advances in Neural Information Processing Systems(NIPS)*.

Shay Cohen and Giorgio Satta. 2013. Approximate PCFG Parsing Using Tensor Decomposition *Proceedings of NAACL-HLT*, 487–496.

Michael Collins. 2002. Discriminative training methods for hidden Markov Models: Theory and Experiments with Perceptron. Algorithms *Proceedings of Empirical Methods in Natural Language Processing(EMNLP)*, 10:1–8.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Schwartz, and Yoram Singer. 2006. Online Passive-Aggressive Algorithms *Journal of Machine Learning Research(JMLR)*, 7:551–585.

Maryam Fazel. 2002. Matrix Rank Minimization with Applications *PhD thesis, Stanford University*.

Kevin Gimpel, and Noah A. Smith 2012. Structured Ramp Loss Minimization for Machine Translation *Proceedings of North American Chapter of the Association for Computational Linguistics(NAACL)*, 221-231.

Tamir Hazan, Simon Polak, and Amnon Shashua 2005. Sparse Image Coding using a 3D Non-negative Tensor Factorization *Proceedings of the International Conference on Computer Vision (ICCV)*.

Mark Hopkins and Jonathan May. 2011. Tuning as Reranking *Proceedings of Empirical Methods in Natural Language Processing(EMNLP)*, 1352-1362.

Tamara Kolda and Brett Bader. 2009. Tensor Decompositions and Applications *SIAM Review*, 51:455-550.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online Large-Margin Training of Dependency Parsers *Proceedings of the 43rd Annual Meeting of the ACL*, 91–98.

Amnon Shashua, and Tamir Hazan. 2005. Non-Negative Tensor Factorization with Applications to Statistics and Computer Vision *Proceedings of the International Conference on Machine Learning (ICML)*.

Tim Van de Cruys, Thierry Poibeau, and Anna Korhonen. 2013. A Tensor-based Factorization Model of Semantic Compositionality *Proceedings of NAACL-HLT*, 1142–1151.

A Proof of Theorem 1

Proof. For $D = 1$, it is obvious that if a set of real numbers $\{x_1, \ldots, x_n\}$ can be represented by a rank-1 matrix, it can always be represented by a vector, but the reverse is not true.

For $D > 1$, if $\{x_1, \ldots, x_n\}$ can be represented by $\mathcal{P} = \mathbf{p}_1 \otimes \mathbf{p}_2 \otimes \ldots \otimes \mathbf{p}_D$, namely $x_i = \mathcal{P}_{i_1,\ldots,i_D} = \prod_{d=1}^{D} p^d_{i_d}$, then for any component vector in mode d,

$$[p^d_1, p^d_2, \ldots, p^d_{n_d}] = [s^d_1 p^d_1, s^d_2 p^d_1, \ldots, s^d_{n^p_d} p^d_1]$$

where n^p_d is the size of mode d of $\mathcal{P}$, s^d_j is a constant and $s^d_j = \frac{p_{i1,\ldots,i_{d-1},j,i_{d+1},\ldots,i_D}}{p_{i1,\ldots,i_{d-1},1,i_{d+1},\ldots,i_D}}$ Therefore

$$x_i = \mathcal{P}_{i_1,\ldots,i_D} = x_{1,\ldots,1} \prod_{d=1}^{D} s^d_{i_d} \tag{7}$$

and this representation is unique for a given D(up to the ordering of $\mathbf{p}_j$ and s^d_j in $\mathbf{p}_j$, which simply assigns $\{x_1, \ldots, x_n\}$ with different indices in the tensor), due to the pairwise proportional constraint imposed by $x_i/x_j, i, j = 1, \ldots, n$.

If x_i can also be represented by $\mathcal{Q}$, then $x_i = \mathcal{Q}_{i_1,\ldots,i_{D+1}} = x_{1,\ldots,1} \prod_{d=1}^{D+1} t^d_{i_d}$, where t^d_j has a similar definition as s^d_j. Then it must be the case that

$$\begin{aligned} &\exists d_1, d_2 \in \{1, \ldots, D+1\}, d \in \{1, \ldots, D\}, d_1 \neq d_2 \\ &s.t. \\ &t^{d_1}_{i_{d_1}} t^{d_2}_{i_{d_2}} = s^d_{i_d}, \\ &t^{d_a}_{i_{d_a}} = s^{d_b}_{i_{d_b}}, \quad d_a \neq d_1, d_2, \quad d_b \neq d \end{aligned} \tag{8}$$

since otherwise $\{x_1, \ldots, x_n\}$ would be represented by a different set of factors than those given in Equation (7).

Therefore, in order for tensor $\mathcal{Q}$ to represent the same set of real numbers that $\mathcal{P}$ represents, there needs to exist a vector $[s^d_1, \ldots, s^d_{n_d}]$ that can be represented by a rank-1 matrix as indicated by Equation (8), which is in general not guaranteed.

On the other hand, if $\{x_1, \ldots, x_n\}$ can be represented by $\mathcal{Q}$, namely

$$x_i = \mathcal{Q}_{i_1,\ldots,i_{D+1}} = \prod_{d=1}^{D+1} q^d_{i_d}$$

then we can just pick $d_1 \in \{1, \ldots, D\}, d_2 = d_1 + 1$ and let

$$\mathbf{q}' = [q^{d_1}_1 q^{d_2}_1, q^{d_1}_1 q^{d_2}_2, \ldots, q^{d_1}_{n^q_{d_2}} q^{d_2}_{n^q_{d_1}}]$$

and

$$\mathcal{Q}' = \mathbf{q}_1 \otimes \ldots \otimes \mathbf{q}_{d_1-1} \otimes \mathbf{q}' \otimes \mathbf{q}_{d_2+1} \otimes \ldots \otimes \mathbf{q}_{D+1}$$

Hence $\{x_1, \ldots, x_n\}$ can also be represented by a D^{th} order tensor $\mathcal{Q}'$. □

Graph-based Semi-Supervised Learning of Translation Models from Monolingual Data

Avneesh Saluja*
Carnegie Mellon University
Pittsburgh, PA 15213, USA
avneesh@cs.cmu.edu

Hany Hassan, Kristina Toutanova, Chris Quirk
Microsoft Research
Redmond, WA 98502, USA
hanyh,kristout,chrisq@microsoft.com

Abstract

Statistical phrase-based translation learns translation rules from bilingual corpora, and has traditionally only used monolingual evidence to construct features that rescore existing translation candidates. In this work, we present a semi-supervised graph-based approach for generating new translation rules that leverages bilingual and monolingual data. The proposed technique first constructs phrase graphs using both source and target language monolingual corpora. Next, graph propagation identifies translations of phrases that were not observed in the bilingual corpus, assuming that similar phrases have similar translations. We report results on a large Arabic-English system and a medium-sized Urdu-English system. Our proposed approach significantly improves the performance of competitive phrase-based systems, leading to consistent improvements between 1 and 4 BLEU points on standard evaluation sets.

1 Introduction

Statistical approaches to machine translation (SMT) use sentence-aligned, parallel corpora to learn translation rules along with their probabilities. With large amounts of data, phrase-based translation systems (Koehn et al., 2003; Chiang, 2007) achieve state-of-the-art results in many typologically diverse language pairs (Bojar et al., 2013). However, the limiting factor in the success of these techniques is parallel data availability. Even in resource-rich languages, learning reliable translations of multiword phrases is a challenge, and an adequate phrasal inventory is crucial for effective translation. This problem is exacerbated in the many language pairs for which parallel resources are either limited or nonexistent. While parallel data is generally scarce, monolingual resources exist in abundance and are being created at accelerating rates. Can we use monolingual data to augment the phrasal translations acquired from parallel data?

The challenge of learning translations from monolingual data is of long standing interest, and has been approached in several ways (Rapp, 1995; Callison-Burch et al., 2006; Haghighi et al., 2008; Ravi and Knight, 2011). Our work introduces a new take on the problem using graph-based semi-supervised learning to acquire translation rules and probabilities by leveraging both monolingual and parallel data resources. On the source side, labeled phrases (those with known translations) are extracted from bilingual corpora, and unlabeled phrases are extracted from monolingual corpora; together they are embedded as nodes in a graph, with the monolingual data determining edge strengths between nodes (§2.2). Unlike previous work (Irvine and Callison-Burch, 2013a; Razmara et al., 2013), we use higher order n-grams instead of restricting to unigrams, since our approach goes beyond OOV mitigation and can enrich the entire translation model by using evidence from monolingual text. This enhancement alone results in an improvement of almost 1.4 BLEU points. On the target side, phrases initially consisting of translations from the parallel data are selectively expanded with generated candidates (§2.1), and are embedded in a target graph.

We then limit the set of translation options for each unlabeled source phrase (§2.3), and using a structured graph propagation algorithm, where translation information is propagated from labeled to unlabeled phrases proportional to *both* source and target phrase similarities, we estimate probability distributions over translations for

*This work was done while the first author was interning at Microsoft Research

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 676–686,
Baltimore, Maryland, USA, June 23-25 2014.

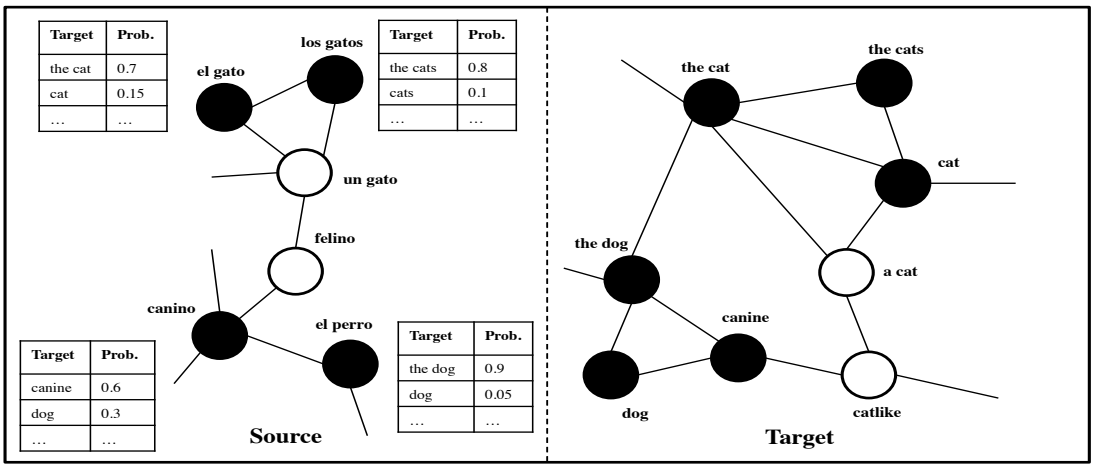


Figure 1: Example source and target graphs used in our approach. Labeled phrases on the source side are black (with their corresponding translations on the target side also black); unlabeled and generated (§2.1) phrases on the source and target sides respectively are white. Labeled phrases also have conditional probability distributions defined over target phrases, which are extracted from the parallel corpora.

the unlabeled source phrases (§2.4). The additional phrases are incorporated in the SMT system through a secondary phrase table (§2.5). We evaluated the proposed approach on both Arabic-English and Urdu-English under a range of scenarios (§3), varying the amount and type of monolingual corpora used, and obtained improvements between 1 and 4 BLEU points, even when using very large language models.

2 Generation & Propagation

Our goal is to obtain translation distributions for source phrases that are not present in the phrase table extracted from the parallel corpus. Both parallel and monolingual corpora are used to obtain these probability distributions over target phrases. We assume that sufficient parallel resources exist to learn a basic translation model using standard techniques, and also assume the availability of larger monolingual corpora in both the source and target languages. Although our technique applies to phrases of any length, in this work we concentrate on unigram and bigram phrases, which provides substantial computational cost savings.

Monolingual data is used to construct *separate* similarity graphs over phrases (word sequences), as illustrated in Fig. 1. The source similarity graph consists of phrase nodes representing sequences of words in the source language. If a source phrase is found in the baseline phrase table it is called a **labeled** phrase: its conditional empirical probability distribution over target phrases (estimated from the parallel data) is used as the label, and is subsequently never changed. Otherwise it is called an **unlabeled** phrase, and our algorithm finds labels (translations) for these unlabeled phrases, with the help of the graph-based representation. The label space is thus the phrasal translation inventory, and like the source side it can also be represented in terms of a graph, initially consisting of target phrase nodes from the parallel corpus.

For the unlabeled phrases, the set of possible target translations could be extremely large (e.g., all target language n-grams). Therefore, we first **generate** and fix a list of possible target translations for each unlabeled source phrase. We then **propagate** by deriving a probability distribution over these target phrases using graph propagation techniques. Next, we will describe the generation, graph construction and propagation steps.

2.1 Generation

The objective of the generation step is to populate the target graph with *additional* target phrases for all unlabeled source phrases, yielding the full set of possible translations for the phrase. Prior to generation, one phrase node for each target phrase occurring in the baseline phrase table is added to the target graph (black nodes in Fig. 1's target graph). We only consider target phrases whose source phrase is a bigram, but it is worth noting that the target phrases are of variable length.

The generation component is based on the observation that for structured label spaces, such as translation candidates for source phrases in SMT, even similar phrases have slightly different labels (target translations). The exponential dependence

of the sizes of these spaces on the length of instances is to blame. Thus, the target phrase inventory from the parallel corpus may be inadequate for unlabeled instances. We therefore need to enrich the target or label space for unknown phrases. A naïve way to achieve this goal would be to extract all n-grams, from $n = 1$ to a maximum n-gram order, from the monolingual data, but this strategy would lead to a combinatorial explosion in the number of target phrases.

Instead, by intelligently expanding the target space using linguistic information such as morphology (Toutanova et al., 2008; Chahuneau et al., 2013), or relying on the baseline system to generate candidates similar to self-training (McClosky et al., 2006), we can tractably propose novel translation candidates (white nodes in Fig. 1's target graph) whose probabilities are then estimated during propagation. We refer to these additional candidates as "generated" candidates.

To generate new translation candidates using the baseline system, we decode each unlabeled source bigram to generate its m-best translations. This set of candidate phrases is filtered to include only n-grams occurring in the target monolingual corpus, and helps to prune passed-through OOV words and invalid translations. To generate new translation candidates using morphological information, we morphologically segment words into prefixes, stem, and suffixes using linguistic resources. We assume that a morphological analyzer which provides context-independent analysis of word types exists, and implements the functions $\text{STEM}(f)$ and $\text{STEM}(e)$ for source and target word types. Based on these functions, source and target sequences of words can be mapped to sequences of stems. The morphological generation step adds to the target graph all target word sequences from the monolingual data that map to the same stem sequence as one of the target phrases occurring in the baseline phrase table. In other words, this step adds phrases that are morphological variants of existing phrases, differing only in their affixes.

2.2 Graph Construction

At this stage, there exists a list of source bigram phrases, both labeled and unlabeled, as well as a list of target language phrases of variable length, originating from both the phrase table and the generation step. To determine pairwise phrase similarities in order to embed these nodes in their graphs, we utilize the monolingual corpora on both the source and target sides to extract distributional features based on the context surrounding each phrase. For a phrase, we look at the p words before and the p words after the phrase, explicitly distinguishing between the two sides, but not distance (i.e., bag of words on each side). Co-occurrence counts for each feature (context word) are accumulated over the monolingual corpus, and these counts are converted to pointwise mutual information (PMI) values, as is standard practice when computing distributional similarities. Cosine similarity between two phrases' PMI vectors is used for similarity, and we take only the k most similar phrases for each phrase, to create a k-nearest neighbor similarity matrix for both source and target language phrases. These graphs are distinct, in that propagation happens within the two graphs but not between them.

While accumulating co-occurrence counts for each phrase, we also maintain an inverted index data structure, which is a mapping from features (context words) to phrases that co-occur with that feature within a window of p.[1] The inverted index structure reduces the graph construction cost from $\theta(n^2)$, by only computing similarities for a subset of all possible pairs of phrases, namely other phrases that have at least one feature in common.

2.3 Candidate Translation List Construction

As mentioned previously, we construct and fix a set of translation candidates, i.e., the label set for each unlabeled source phrase. The probability distribution over these translations is estimated through graph propagation, and the probabilities of items outside the list are assumed to be zero.

We obtain these candidates from two sources:[2]

1. The union of each unlabeled phrase's labeled neighbors' labels, which represents the set of target phrases that occur as translations of source phrases that are similar to the unlabeled source phrase. For *un gato* in Fig. 1, this source would yield *the cat* and *cat*, among others, as candidates.

2. The generated candidates for the unlabeled phrase – the ones from the baseline system's

[1] The q most frequent words in the monolingual corpus were removed as keys from this mapping, as these high entropy features do not provide much information.

[2] We also obtained the k-nearest neighbors of the translation candidates generated through these methods by utilizing the target graph, but this had minimal impact.

decoder output, or from a morphological generator (e.g., *a cat* and *catlike* in Fig. 1).

The morphologically-generated candidates for a given source unlabeled phrase are initially defined as the target word sequences in the monolingual data that have the same stem sequence as one of the baseline's target translations for a source phrase which has the same stem sequence as the unlabeled source phrase. These candidates are scored using stem-level translation probabilities, morpheme-level lexical weighting probabilities, and a language model, and only the top 30 candidates are included.

After obtaining candidates from these two possible sources, the list is sorted by forward lexical score, using the lexical models of the baseline system. The top r candidates are then chosen for each phrase's translation candidate list.

In Figure 2 we provide example outputs of our system for a handful of unlabeled source phrases, and explicitly note the source of the translation candidate ('G' for generated, 'N' for labeled neighbor's label).

2.4 Graph Propagation

A graph propagation algorithm transfers label information from labeled nodes to unlabeled nodes by following the graph's structure. In some applications, a label may consist of class membership information, e.g., each node can belong to one of a certain number of classes. In our problem, the "label" for each node is actually a probability distribution over a set of translation candidates (target phrases). For a given node f, let e refer to a candidate in the label set for node f; then in graph propagation, the probability of candidate e given source phrase f in iteration $t+1$ is:

$$\mathbb{P}^{t+1}(e|f) = \sum_{j \in \mathcal{N}(f)} T_s(j|f)\mathbb{P}^t(e|j) \quad (1)$$

where the set $\mathcal{N}(f)$ contains the (labeled and unlabeled) neighbors of node f, and $T_s(j|f)$ is a term that captures how similar nodes f and j are. This quantity is also known as the propagation probability, and its exact form will depend on the type of graph propagation algorithm used. For our purposes, node f is a source phrasal node, the set $\mathcal{N}(f)$ refers to other source phrases that are neighbors of f (restricted to the k-nearest neighbors as in §2.2), and the aim is to estimate $P(e|f)$, the probability of target phrase e being a phrasal translation of source phrase f.

A classic propagation algorithm that has been suitably modified for use in bilingual lexicon induction (Tamura et al., 2012; Razmara et al., 2013) is the **label propagation** (LP) algorithm of Zhu et al. (2003). In this case, $T_s(f, j)$ is chosen to be:

$$T_s(j|f) = \frac{w^s_{f,j}}{\sum_{j' \in \mathcal{N}(f)} w^s_{f,j'}} \quad (2)$$

where $w^s_{f,j}$ is the cosine similarity (as computed in §2.2) between phrase f and phrase j on side s (the source side).

As evident in Eq. 2, LP only takes into account source language similarity of phrases. To see this observation more clearly, let us reformulate Eq. 1 more generally as:

$$\mathbb{P}^{t+1}(e|f) = \sum_{j \in \mathcal{N}(f)} T_s(j|f) \sum_{e' \in \mathcal{H}(j)} T_t(e'|e)\mathbb{P}^t(e'|j) \quad (3)$$

where $\mathcal{H}(j)$ is the translation candidate set for source phrase j, and $T_t(e'|e)$ is the propagation probability between nodes or phrases e and e' on the *target* side. We have simply replaced $\mathbb{P}^t(e|j)$ with $\sum_{e' \in \mathcal{H}(j)} T_t(e'|e)\mathbb{P}^t(e'|j)$, defining it in terms of j's translation candidate list.

Note that in the original LP formulation the target side information is disregarded, i.e., $T_t(e'|e) = 1$ if and only if $e = e'$ and 0 otherwise. As a result, LP is suboptimal for our needs, since it is unable to appropriately handle generated translation candidates for the unlabeled phrases. These translation candidates are usually not present as translations for the labeled phrases (or for the labeled phrases that neighbor the unlabeled one in question). When propagating information from the labeled phrases, such candidates will obtain no probability mass since $e \neq e'$. Thus, due to the setup of the problem, LP naturally biases away from translation candidates produced during the generation step (§2.1).

2.4.1 Structured Label Propagation

The label set we are considering has a similarity structure encoded by the target graph. How can we exploit this structure in graph propagation on the source graph? In Liu *et al.* (2012), the authors generalize label propagation to **structured label propagation** (SLP) in an effort to work more elegantly with structured labels. In particular, the definition of target similarity is similar to that of source similarity:

$$T_t(e'|e) = \frac{w^t_{e,e'}}{\sum_{e'' \in \mathcal{H}(j)} w^t_{e,e''}} \quad (4)$$

Therefore, the final update equation in SLP is:

$$\mathbb{P}^{t+1}(e|f) = \sum_{j \in \mathcal{N}(f)} T_s(j|f) \sum_{e' \in \mathcal{H}(j)} T_t(e'|e)\mathbb{P}^t(e'|j) \quad (5)$$

With this formulation, even if $e \neq e'$, the similarity $T_t(e'|e)$ as determined by the target phrase graph will dictate propagation probability. We renormalize the probability distributions after each propagation step to sum to one over the fixed list of translation candidates, and run the SLP algorithm to convergence.[3]

2.5 Phrase-based SMT Expansion

After graph propagation, each unlabeled phrase is labeled with a categorical distribution over the set of translation candidates defined in §2.3. In order to utilize these newly acquired phrase pairs, we need to compute their relevant features. The phrase pairs have four log-probability features with two likelihood features and two lexical weighting features. In addition, we use a sophisticated lexicalized hierarchical reordering model (HRM) (Galley and Manning, 2008) with five features for each phrase pair.

We utilize the graph propagation-estimated forward phrasal probabilities $\mathbb{P}(e|f)$ as the forward likelihood probabilities for the acquired phrases; to obtain the backward phrasal probability for a given phrase pair, we make use of Bayes' Theorem:

$$\mathbb{P}(f|e) = \frac{\mathbb{P}(e|f)\mathbb{P}(f)}{\mathbb{P}(e)}$$

where the marginal probabilities of source and target phrases e and f are obtained from the counts extracted from the monolingual data. The baseline system's lexical models are used for the forward and backward lexical scores. The HRM probabilities for the new phrase pairs are estimated from the baseline system by backing-off to the average values for phrases with similar length.

3 Evaluation

We performed an extensive evaluation to examine various aspects of the approach along with overall system performance. Two language pairs were used: Arabic-English and Urdu-English. The Arabic-English evaluation was used to validate the decisions made during the development of our method and also to highlight properties of the technique. With it, in §3.2 we first analyzed the impact of utilizing phrases instead of words and SLP instead of LP; the latter experiment underscores the importance of generated candidates. We also look at how adding morphological knowledge to the generation process can further enrich performance. In §3.3, we then examined the effect of using a very large 5-gram language model training on 7.5 billion English tokens to understand the nature of the improvements in §3.2. The Urdu to English evaluation in §3.4 focuses on how noisy parallel data and completely monolingual (i.e., not even comparable) text can be used for a realistic low-resource language pair, and is evaluated with the larger language model only. We also examine how our approach can learn from noisy parallel data compared to the traditional SMT system.

Baseline phrasal systems are used both for comparison and for generating translation candidates for unlabeled phrases as described in §2.1. The baseline is a state-of-the-art phrase-based system; we perform word alignment using a lexicalized hidden Markov model, and then the phrase table is extracted using the `grow-diag-final` heuristic (Koehn et al., 2003). The 13 baseline features (2 lexical, 2 phrasal, 5 HRM, and 1 language model, word penalty, phrase length feature and distortion penalty feature) were tuned using MERT (Och, 2003), which is also used to tune the 4 feature weights introduced by the secondary phrase table (2 lexical and 2 phrasal, other features being shared between the two tables). For all systems, we use a distortion limit of 4. We use case-insensitive BLEU (Papineni et al., 2002) to evaluate translation quality.

3.1 Datasets

Bilingual corpus statistics for both language pairs are presented in Table 2. For Arabic-English, our training corpus consisted of 685k sentence pairs from standard LDC corpora[4]. The NIST MT06 and MT08 Arabic-English evaluation sets (combining the newswire and weblog domains for both sets), with four references each, were used as tuning and testing sets respectively. For Urdu-English, the training corpus was provided by the LDC for the NIST Urdu-English MT evaluation, and most of the data was automatically acquired from the web, making it quite noisy. After filtering, there are approximately 65k parallel sen-

[3] Empirically within a few iterations and a wall-clock time of less than 10 minutes in total.

[4] LDC2007T08 and LDC2008T09

Parameter	Description	Value
m	m-best candidate list size when bootstrapping candidates in generation stage.	100
p	Window size on each side when extracting features for phrases.	2
q	Filter the q most frequent words when storing the inverted index data structure for graph construction. Both source and target sides share the same value.	25
k	Number of neighbors stored for each phrase for both source and target graphs. This parameter controls the sparsity of the graph.	500
r	Maximum size of translation candidate list for unlabeled phrases.	20

Table 1: Parameters, explanation of their function, and value chosen.

tences; these were supplemented by an additional 100k dictionary entries. Tuning and test data consisted of the MT08 and MT09 evaluation corpora, once again a mixture of news and web text.

Corpus	Sentences	Words (Src)
Ar-En Train	685,502	17,055,168
Ar-En Tune (MT06)	1,664	33,739
Ar-En Test (MT08)	1,360	42,472
Ur-En Train	165,159	1,169,367
Ur-En Tune (MT08)	1,864	39,925
Ur-En Test (MT09)	1,792	39,922

Table 2: Bilingual corpus statistics for the Arabic-English and Urdu-English datasets used.

Table 3 contains statistics for the monolingual corpora used in our experiments. From these corpora, we extracted all sentences that contained at least one source or target phrase match to compute features for graph construction. For the Arabic to English experiments, the monolingual corpora are taken from the AFP Arabic and English Gigaword corpora and are of a similar date range to each other (1994-2010), rendering them comparable but not sentence-aligned or parallel.

Corpus	Sentences	Words
Ar Comparable	10.2m	290m
En I Comparable	29.8m	900m
Ur Noisy Parallel	470k	5m
En II Noisy Parallel	470k	4.7m
Ur Non-Comparable	7m	119m
En II Non-Comparable	17m	510m

Table 3: Monolingual corpus statistics for the Arabic-English and Urdu-English evaluations. The monolingual corpora can be sub-divided into comparable, noisy parallel, and non-comparable components. En I refers to the English side of the Arabic-English corpora, and En II to the English side of the Urdu-English corpora.

For the Urdu-English experiments, completely non-comparable monolingual text was used for graph construction; we obtained the Urdu side through a web-crawler, and a subset of the AFP Gigaword English corpus was used for English. In addition, we obtained a corpus from the ELRA[5], which contains a mix of parallel and monolingual data; based on timestamps, we extracted a comparable English corpus for the ELRA Urdu monolingual data to form a roughly 470k-sentence "noisy parallel" set. We used this set in two ways: either to augment the parallel data presented in Table 2, or to augment the non-comparable monolingual data in Table 3 for graph construction.

For the parameters introduced throughout the text, we present in Table 1 a reminder of their interpretation as well as the values used in this work.

3.2 Experimental Variations

In our first set of experiments, we looked at the impact of choosing bigrams over unigrams as our basic unit of representation, along with performance of LP (Eq. 2) compared to SLP (Eq. 4). Recall that LP only takes into account source similarity; since the vast majority of generated candidates do not occur as labeled neighbors' labels, restricting propagation to the source graph drastically reduces the usage of generated candidates as labels, but does not completely eliminate it. In these experiments, we utilize a reasonably-sized 4-gram language model trained on 900m English tokens, i.e., the English monolingual corpus.

Table 4 presents the results of these variations; overall, by taking into account generated candidates appropriately and using bigrams ("SLP 2-gram"), we obtained a 1.13 BLEU gain on the test set. Using unigrams ("SLP 1-gram") actually does worse than the baseline, indicating the importance of focusing on translations for sparser bigrams. While LP ("LP 2-gram") does reasonably well, its underperformance compared to SLP underlines the importance of enriching the translation space with generated candidates and handling these candidates appropriately.[6] In "SLP-

[5]ELRA-W0038

[6]It is relatively straightforward to combine both unigrams and bigrams in one source graph, but for experimental clarity we did not mix these phrase lengths.

HalfMono", we use only half of the monolingual comparable corpora, and still obtain an improvement of 0.56 BLEU points, indicating that adding more monolingual data is likely to improve the system further. Interestingly, biasing away from generated candidates using all the monolingual data ("LP 2-gram") performs similarly to using half the monolingual corpora and handling generated candidates properly ("SLP-HalfMono").

	BLEU	
Setup	**Tune**	**Test**
Baseline	39.33	38.09
SLP 1-gram	39.47	37.85
LP 2-gram	40.75	38.68
SLP 2-gram	41.00	39.22
SLP-HalfMono 2-gram	40.82	38.65
SLP+Morph 2-gram	41.02	39.35

Table 4: Results for the Arabic-English evaluation. The LP vs. SLP comparison highlights the importance of target side enrichment via translation candidate generation, 1-gram vs. 2-gram comparisons highlight the importance of emphasizing phrases, utilizing half the monolingual data shows sensitivity to monolingual corpus size, and adding morphological information results in additional improvement.

Additional morphologically generated candidates were added in this experiment as detailed in §2.3. We used a simple hand-built Arabic morphological analyzer that segments word types based on regular expressions, and an English lexicon-based morphological analyzer. The morphological candidates add a small amount of improvement, primarily by targeting genuine OOVs.

3.3 Large Language Model Effect

In this set of experiments, we examined if the improvements in §3.2 can be explained primarily through the extraction of language model characteristics during the semi-supervised learning phase, or through orthogonal pieces of evidence. Would the improvement be less substantial had we used a very large language model?

To answer this question we trained a 5-gram language model on 570M sentences (7.6B tokens), with data from various sources including the Gigaword corpus[7], WMT and European Parliamentary Proceedings[8], and web-crawled data from Wikipedia and the web. Only m-best generated candidates from the baseline were considered during generation, along with labeled neighbors' labels.

[7] LDC2011T07

[8] http://www.statmt.org/wmt13/

	BLEU	
Setup	**Tune**	**Test**
Baseline+LargeLM	41.48	39.86
SLP+LargeLM	42.82	41.29

Table 5: Results with the large language model scenario. The gains are even better than with the smaller language model.

Table 5 presents the results of using this language model. We obtained a robust, 1.43-BLEU point gain, indicating that the addition of the newly induced phrases provided genuine translation improvements that cannot be compensated by the language model effect. Further examination of the differences between the two systems yielded that most of the improvements are due to better bigrams and trigrams, as indicated by the breakdown of the BLEU score precision per n-gram, and primarily leverages higher quality generated candidates from the baseline system. We analyze the output of these systems further in the output analysis section below (§3.5).

3.4 Urdu-English

In order to evaluate the robustness of these results beyond one language pair, we looked at Urdu-English, a low resource pair likely to benefit from this approach. In this set of experiments, we used the large language model in §3.3, and only used baseline-generated candidates. We experimented with two extreme setups that differed in the data assumed parallel, from which we built our baseline system, and the data treated as monolingual, from which we built our source and target graphs.

In the first setup, we use the noisy parallel data for graph construction and augment the non-comparable corpora with it:

- parallel: "Ur-En Train"
- Urdu monolingual: "Ur Noisy Parallel"+"Ur Non-Comparable"
- English monolingual: "En II Noisy Parallel"+"En II Non-Comparable"

The results from this setup are presented as "Baseline" and "SLP+Noisy" in Table 6. In the second setup, we train a baseline system using the data in Table 2, augmented with the noisy parallel text:

- parallel: "Ur-En Train"+"Ur Noisy Parallel"+"En II Noisy Parallel"
- Urdu monolingual: "Ur Non-Comparable"
- English monolingual: "En II Non-Comparable"

Ex	Source	Reference	Baseline	System
1 (Ar)	ارسال التعزيزات	sending reinforcements	strong reinforcements	sending reinforcements (N)
2 (Ar)	ل+ الاندثار	with extinction	OOV	with extinction (N)
3 (Ar)	تحبط محاولة	thwarts	address	thwarted (N)
4 (Ar)	نسبت الي	was quoted as saying	attributed to	was quoted as saying (G)
5 (Ar)	أوضح عبد المحمود	abdalmahmood said	he said abdul mahmood	mahmood said (G)
6 (Ar)	تراه منكبا	it deems	OOV	it deems (G)
7 (Ur)	پر امید	I am hopeful	this hope	I am hopeful (N)
8 (Ur)	اپنا دفاع	to defend him	to defend	to defend himself (G)
9 (Ur)	گفتگو کی۔	while speaking	In the	in conversation (N)

Figure 2: Nine example outputs of our system vs. the baseline highlighting the properties of our approach. Each example is labeled (Ar) for Arabic source or (Ur) for Urdu source, and system candidates are labeled with (N) if the candidate unlabeled phrase's labeled neighbor's label, or (G) if the candidate was generated.

The results from this setup are presented as "Baseline+Noisy" and "SLP" in Table 6. The two setups allow us to examine how effectively our method can learn from the noisy parallel data by treating it as monolingual (i.e., for graph construction), compared to treating this data as parallel, and also examines the realistic scenario of using completely non-comparable monolingual text for graph construction as in the second setup.

	BLEU	
Setup	**Tune**	**Test**
Baseline	21.87	21.17
SLP+Noisy	26.42	25.38
Baseline+Noisy	27.59	27.24
SLP	28.53	28.43

Table 6: Results for the Urdu-English evaluation evaluated with BLEU. All experiments were conducted with the larger language model, and generation only considered the m-best candidates from the baseline system.

In the first setup, we get a huge improvement of 4.2 BLEU points ("SLP+Noisy") when using the monolingual data and the noisy parallel data for graph construction. Our method obtained much of the gains achieved by the supervised baseline approach that utilizes the noisy parallel data in conjunction with the NIST-provided parallel data ("Baseline+Noisy"), but with fewer assumptions on the nature of the corpora (monolingual vs. parallel). Furthermore, despite completely unaligned, non-comparable monolingual text on the Urdu and English sides, and a very large language model, we can still achieve gains in excess of 1.2 BLEU points ("SLP") in a difficult evaluation scenario, which shows that the technique adds a genuine translation improvement over and above naïve memorization of n-gram sequences.

3.5 Analysis of Output

Figure 2 looks at some of the sample hypotheses produced by our system and the baseline, along with reference translations. The outputs produced by our system are additionally annotated with the origin of the candidate, i.e., labeled neighbor's label (N) or generated (G).

The Arabic-English examples are numbered 1 to 5. The first example shows a source bigram unknown to the baseline system, resulting in a suboptimal translation, while our system proposes the correct translation of "sending reinforcements". The second example shows a word that was an OOV for the baseline system, while our system got a perfect translation. The third and fourth examples represent bigram phrases with much better translations compared to backing off to the lexical translations as in the baseline. The fifth Arabic-English example demonstrates the pitfalls of over-reliance on the distributional hypothesis: the source bigram corresponding to the name "abd almahmood" is distributional similar to another named entity "mahmood" and the English equivalent is offered as a translation. The distributional hypothesis can sometimes be misleading. The sixth example shows how morphological information can propose novel candidates: an OOV word is broken down to its stem via the analyzer and candidates are generated based on the stem.

The Urdu-English examples are numbered 7 to 9. In example 7, the bigram "par umeed" (corresponding to "hopeful") is never seen in the baseline system, which has only seen "umeed" ("hope"). By leveraging the monolingual corpus to understand the context of this unlabeled bigram, we can utilize the graph structure to propose a syntactically correct form, also resulting in a more fluent and correct sentence as determined by the language model. Examples 8 & 9 show cases where the baseline deletes words or translates them into more common words e.g., "conversation" to "the", while our system proposes reasonable candidates.

4 Related Work

The idea presented in this paper is similar in spirit to bilingual lexicon induction (BLI), where a seed lexicon in two different languages is expanded with the help of monolingual corpora, primarily by extracting distributional similarities from the data using word context. This line of work, initiated by Rapp (1995) and continued by others (Fung and Yee, 1998; Koehn and Knight, 2002) (*inter alia*) is limited from a downstream perspective, as translations for only a small number of words are induced and oftentimes for common or frequently occurring ones only. Recent improvements to BLI (Tamura et al., 2012; Irvine and Callison-Burch, 2013b) have contained a graph-based flavor by presenting label propagation-based approaches using a seed lexicon, but evaluation is once again done on top-1 or top-3 accuracy, and the focus is on unigrams.

Razmara et al. (2013) and Irvine and Callison-Burch (2013a) conduct a more extensive evaluation of their graph-based BLI techniques, where the emphasis and end-to-end BLEU evaluations concentrated on OOVs, i.e., unigrams, and not on enriching the entire translation model. As with previous BLI work, these approaches only take into account source-side similarity of words; only moderate gains (and in the latter work, on a subset of language pairs evaluated) are obtained. Additionally, because of our structured propagation algorithm, our approach is better at handling multiple translation candidates and does not need to restrict itself to the top translation.

Klementiev et al. (2012) propose a method that utilizes a pre-existing phrase table and a small bilingual lexicon, and performs BLI using monolingual corpora. The operational scope of their approach is limited in that they assume a scenario where unknown phrase pairs are provided (thereby sidestepping the issue of translation candidate generation for completely unknown phrases), and what remains is the estimation of phrasal probabilities. In our case, we obtain the phrase pairs from the graph structure (and therefore indirectly from the monolingual data) and a separate generation step, which plays an important role in good performance of the method. Similarly, Zhang and Zong (2013) present a series of heuristics that are applicable in a fairly narrow setting.

The notion of translation consensus, wherein similar sentences on the source side are encouraged to have similar target language translations, has also been explored via a graph-based approach (Alexandrescu and Kirchhoff, 2009). Liu et al. (2012) extend this method by proposing a novel structured label propagation algorithm to deal with the generalization of propagating *sets* of labels instead of single labels, and also integrated information from the graph into the decoder. In fact, we utilize this algorithm in our propagation step (§2.4). However, the former work operates only at the level of sentences, and while the latter does extend the framework to sub-spans of sentences, they do not discover new translation pairs or phrasal probabilities for new pairs at all, but instead re-estimate phrasal probabilities using the graph structure and add this score as an additional feature during decoding.

The goal of leveraging non-parallel data in machine translation has been explored from several different angles. Paraphrases extracted by "pivoting" via a third language (Callison-Burch et al., 2006) can be derived solely from monolingual corpora using distributional similarity (Marton et al., 2009). Snover et al. (2008) use cross-lingual information retrieval techniques to find potential sentence-level translation candidates among comparable corpora. In this case, the goal is to try and construct a corpus as close to parallel as possible from comparable corpora, and is a fairly different take on the problem we are looking at. Decipherment-based approaches (Ravi and Knight, 2011; Dou and Knight, 2012) have generally taken a monolingual view to the problem and combine phrase tables through the log-linear model during feature weight training.

5 Conclusion

In this work, we presented an approach that can expand a translation model extracted from a sentence-aligned, bilingual corpus using a large amount of unstructured, monolingual data in both source and target languages, which leads to improvements of 1.4 and 1.2 BLEU points over strong baselines on evaluation sets, and in some scenarios gains in excess of 4 BLEU points. In the future, we plan to estimate the graph structure through other learned, distributed representations.

Acknowledgments

The authors would like to thank Chris Dyer, Arul Menezes, and the anonymous reviewers for their helpful comments and suggestions.

References

Andrei Alexandrescu and Katrin Kirchhoff. 2009. Graph-based learning for statistical machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL-HLT '09, pages 119–127. Association for Computational Linguistics, June.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria, August. Association for Computational Linguistics.

Chris Callison-Burch, Philipp Koehn, and Miles Osborne. 2006. Improved statistical machine translation using paraphrases. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 17–24, New York City, USA, June. Association for Computational Linguistics.

Victor Chahuneau, Eva Schlinger, Noah A. Smith, and Chris Dyer. 2013. Translating into morphologically rich languages with synthetic phrases. In *Proc. of EMNLP*.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228, June.

Qing Dou and Kevin Knight. 2012. Large scale decipherment for out-of-domain machine translation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 266–275. Association for Computational Linguistics, July.

Pascale Fung and Lo Yuen Yee. 1998. An ir approach for translating new words from nonparallel, comparable texts. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, ACL '98, pages 414–420, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michel Galley and Christopher D. Manning. 2008. A simple and effective hierarchical phrase reordering model. EMNLP '08, pages 848–856, Stroudsburg, PA, USA. Association for Computational Linguistics.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proceedings of ACL-08: HLT*, pages 771–779, Columbus, Ohio, June. Association for Computational Linguistics.

Ann Irvine and Chris Callison-Burch. 2013a. Combining bilingual and comparable corpora for low resource machine translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 262–270, Sofia, Bulgaria, August. Association for Computational Linguistics.

Ann Irvine and Chris Callison-Burch. 2013b. Supervised bilingual lexicon induction with multiple monolingual signals. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 518–523, Atlanta, Georgia, June. Association for Computational Linguistics.

Alexandre Klementiev, Ann Irvine, Chris Callison-Burch, and David Yarowsky. 2012. Toward statistical machine translation without parallel corpora. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 130–140, Avignon, France, April. Association for Computational Linguistics.

Philipp Koehn and Kevin Knight. 2002. Learning a translation lexicon from monolingual corpora. In *In Proceedings of ACL Workshop on Unsupervised Lexical Acquisition*, pages 9–16.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54, Stroudsburg, PA, USA. Association for Computational Linguistics.

Shujie Liu, Chi-Ho Li, Mu Li, and Ming Zhou. 2012. Learning translation consensus with structured label propagation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 302–310, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yuval Marton, Chris Callison-Burch, and Philip Resnik. 2009. Improved statistical machine translation using monolingually-derived paraphrases. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, EMNLP '09, pages 381–390, Singapore, August. Association for Computational Linguistics.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159, New York City, USA, June. Association for Computational Linguistics.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 160–167, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. pages 311–318.

Reinhard Rapp. 1995. Identifying word translations in non-parallel texts. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, ACL '95.

Sujith Ravi and Kevin Knight. 2011. Deciphering foreign language. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 12–21, Portland, Oregon, USA, June. Association for Computational Linguistics.

Majid Razmara, Maryam Siahbani, Gholamreza Haffari, and Anoop Sarkar. 2013. Graph propagation for paraphrasing out-of-vocabulary words in statistical machine translation. In *Proceedings of the 51st of the Association for Computational Linguistics*, ACL-51, Stroudsburg, PA, USA. Association for Computational Linguistics.

Matthew Snover, Bonnie Dorr, and Richard Schwartz. 2008. Language and translation model adaptation using comparable corpora. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 857–866, Stroudsburg, PA, USA. Association for Computational Linguistics.

Akihiro Tamura, Taro Watanabe, and Eiichiro Sumita. 2012. Bilingual lexicon extraction from comparable corpora using label propagation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 24–36.

Kristina Toutanova, Hisami Suzuki, and Achim Ruopp. 2008. Applying morphology generation models to machine translation. In *Proceedings of ACL-08: HLT*, pages 514–522, Columbus, Ohio, June. Association for Computational Linguistics.

Jiajun Zhang and Chengqing Zong. 2013. Learning a phrase-based translation model from monolingual data with application to domain adaptation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1425–1434, Sofia, Bulgaria, August. Association for Computational Linguistics.

Xiaojin Zhu, Zoubin Ghahramani, and John D. Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the Twentieth International Conference on Machine Learning*, ICML '03, pages 912–919.

Using Discourse Structure Improves Machine Translation Evaluation

Francisco Guzmán **Shafiq Joty** **Lluís Màrquez** and **Preslav Nakov**
ALT Research Group
Qatar Computing Research Institute — Qatar Foundation
{fguzman,sjoty,lmarquez,pnakov}@qf.org.qa

Abstract

We present experiments in using discourse structure for improving machine translation evaluation. We first design two discourse-aware similarity measures, which use all-subtree kernels to compare discourse parse trees in accordance with the Rhetorical Structure Theory. Then, we show that these measures can help improve a number of existing machine translation evaluation metrics both at the segment- and at the system-level. Rather than proposing a single new metric, we show that discourse information is complementary to the state-of-the-art evaluation metrics, and thus should be taken into account in the development of future richer evaluation metrics.

1 Introduction

From its foundations, Statistical Machine Translation (SMT) had two defining characteristics: first, translation was modeled as a generative process *at the sentence-level*. Second, it was purely statistical over words or word sequences and made *little to no use of linguistic information*. Although modern SMT systems have switched to a discriminative log-linear framework, which allows for additional sources as features, it is generally hard to incorporate dependencies beyond a small window of adjacent words, thus making it difficult to use linguistically-rich models.

Recently, there have been two promising research directions for improving SMT and its evaluation: (*a*) by using more structured linguistic information, such as syntax (Galley et al., 2004; Quirk et al., 2005), hierarchical structures (Chiang, 2005), and semantic roles (Wu and Fung, 2009; Lo et al., 2012), and (*b*) by going beyond the sentence-level, e.g., translating at the document level (Hardmeier et al., 2012).

Going beyond the sentence-level is important since sentences rarely stand on their own in a well-written text. Rather, each sentence follows smoothly from the ones before it, and leads into the ones that come afterwards. The logical relationship between sentences carries important information that allows the text to express a meaning as a whole beyond the sum of its separate parts.

Note that sentences can be made of several clauses, which in turn can be interrelated through the same logical relations. Thus, in a coherent text, discourse units (sentences or clauses) are logically connected: the meaning of a unit relates to that of the previous and the following units.

Discourse analysis seeks to uncover this coherence structure underneath the text. Several formal theories of discourse have been proposed to describe the coherence structure (Mann and Thompson, 1988; Asher and Lascarides, 2003; Webber, 2004). For example, the Rhetorical Structure Theory (Mann and Thompson, 1988), or RST, represents text by labeled hierarchical structures called Discourse Trees (DTs), which can incorporate several layers of other linguistic information, e.g., syntax, predicate-argument structure, etc.

Modeling discourse brings together the above research directions (a) and (b), which makes it an attractive goal for MT. This is demonstrated by the establishment of a recent workshop dedicated to Discourse in Machine Translation (Webber et al., 2013), collocated with the 2013 annual meeting of the Association of Computational Linguistics.

The area of discourse analysis for SMT is still nascent and, to the best of our knowledge, no previous research has attempted to use rhetorical structure for SMT or machine translation evaluation. One possible reason could be the unavailability of accurate discourse parsers. However, this situation is likely to change given the most recent advances in automatic discourse analysis (Joty et al., 2012; Joty et al., 2013).

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 687–698,
Baltimore, Maryland, USA, June 23-25 2014.

We believe that the semantic and pragmatic information captured in the form of DTs (*i*) can help develop discourse-aware SMT systems that produce coherent translations, and (*ii*) can yield better MT evaluation metrics. While in this work we focus on the latter, we think that the former is also within reach, and that SMT systems would benefit from preserving the coherence relations in the source language when generating target-language translations.

In this paper, rather than proposing yet another MT evaluation metric, we show that discourse information is complementary to many existing evaluation metrics, and thus should not be ignored. We first design two discourse-aware similarity measures, which use DTs generated by a publicly-available discourse parser (Joty et al., 2012); then, we show that they can help improve a number of MT evaluation metrics at the segment- and at the system-level in the context of the WMT11 and the WMT12 metrics shared tasks (Callison-Burch et al., 2011; Callison-Burch et al., 2012).

These metrics tasks are based on sentence-level evaluation, which arguably can limit the benefits of using global discourse properties. Fortunately, several sentences are long and complex enough to present rich discourse structures connecting their basic clauses. Thus, although limited, this setting is able to demonstrate the potential of discourse-level information for MT evaluation. Furthermore, sentence-level scoring (*i*) is compatible with most translation systems, which work on a sentence-by-sentence basis, (*ii*) could be beneficial to modern MT tuning mechanisms such as PRO (Hopkins and May, 2011) and MIRA (Watanabe et al., 2007; Chiang et al., 2008), which also work at the sentence-level, and (*iii*) could be used for re-ranking n-best lists of translation hypotheses.

2 Related Work

Addressing discourse-level phenomena in machine translation is relatively new as a research direction. Some recent work has looked at anaphora resolution (Hardmeier and Federico, 2010) and discourse connectives (Cartoni et al., 2011; Meyer, 2011), to mention two examples.[1] However, so far the attempts to incorporate discourse-related knowledge in MT have been only moderately successful, at best.

[1]We refer the reader to (Hardmeier, 2012) for an in-depth overview of discourse-related research for MT.

A common argument, is that current automatic evaluation metrics such as BLEU are inadequate to capture discourse-related aspects of translation quality (Hardmeier and Federico, 2010; Meyer et al., 2012). Thus, there is consensus that discourse-informed MT evaluation metrics are needed in order to advance research in this direction. Here we suggest some simple ways to create such metrics, and we also show that they yield better correlation with human judgments.

The field of automatic evaluation metrics for MT is very active, and new metrics are continuously being proposed, especially in the context of the evaluation campaigns that run as part of the Workshops on Statistical Machine Translation (WMT 2008-2012), and NIST Metrics for Machine Translation Challenge (MetricsMATR), among others. For example, at WMT12, 12 metrics were compared (Callison-Burch et al., 2012), most of them new.

There have been several attempts to incorporate syntactic and semantic linguistic knowledge into MT evaluation. For instance, at the syntactic level, we find metrics that measure the structural similarity between shallow syntactic sequences (Giménez and Màrquez, 2007; Popovic and Ney, 2007) or between constituency trees (Liu and Gildea, 2005). In the semantic case, there are metrics that exploit the similarity over named entities and predicate-argument structures (Giménez and Màrquez, 2007; Lo et al., 2012).

In this work, instead of proposing a new metric, we focus on enriching current MT evaluation metrics with discourse information. Our experiments show that many existing metrics can benefit from additional knowledge about discourse structure.

In comparison to the syntactic and semantic extensions of MT metrics, there have been very few attempts to incorporate discourse information so far. One example are the semantics-aware metrics of Giménez and Màrquez (2009) and Comelles et al. (2010), which use the Discourse Representation Theory (Kamp and Reyle, 1993) and tree-based discourse representation structures (DRS) produced by a semantic parser. They calculate the similarity between the MT output and references based on DRS subtree matching, as defined in (Liu and Gildea, 2005), DRS lexical overlap, and DRS morpho-syntactic overlap. However, they could not improve correlation with human judgments, as evaluated on the MetricsMATR dataset.

Compared to the previous work, (*i*) we use a different discourse representation (RST), (*ii*) we compare discourse parses using *all-subtree* kernels (Collins and Duffy, 2001), (*iii*) we evaluate on much larger datasets, for several language pairs and for multiple metrics, and (*iv*) we do demonstrate better correlation with human judgments.

Wong and Kit (2012) recently proposed an extension of MT metrics with a measure of document-level *lexical cohesion* (Halliday and Hasan, 1976). Lexical cohesion is achieved using word repetitions and semantically similar words such as synonyms, hypernyms, and hyponyms. For BLEU and TER, they observed improved correlation with human judgments on the MTC4 dataset when linearly interpolating these metrics with their *lexical cohesion* score. Unlike their work, which measures lexical cohesion at the document-level, here we are concerned with *coherence (rhetorical) structure*, primarily at the sentence-level.

3 Our Discourse-Based Measures

Our working hypothesis is that the similarity between the discourse structures of an automatic and of a reference translation provides additional information that can be valuable for evaluating MT systems. In particular, we believe that good translations should tend to preserve discourse relations.

As an example, consider the three discourse trees (DTs) shown in Figure 1: (*a*) for a reference (human) translation, and (*b*) and (*c*) for translations of two different systems on the WMT12 test dataset. The leaves of a DT correspond to contiguous atomic text spans, called *Elementary Discourse Units* or EDUs (three in Figure 1a). Adjacent spans are connected by certain coherence relations (e.g., *Elaboration*, *Attribution*), forming larger discourse units, which in turn are also subject to this relation linking. Discourse units linked by a relation are further distinguished based on their relative importance in the text: *nuclei* are the core parts of the relation while *satellites* are supportive ones. Note that the nuclearity and relation labels in the reference translation are also realized in the system translation in (b), but not in (c), which makes (b) a better translation compared to (c), according to our hypothesis. We argue that existing metrics that only use lexical and syntactic information cannot distinguish well between (b) and (c).

In order to develop a discourse-aware evaluation metric, we first generate discourse trees for the reference and the system-translated sentences using a discourse parser, and then we measure the similarity between the two discourse trees. We describe these two steps below.

3.1 Generating Discourse Trees

In Rhetorical Structure Theory, discourse analysis involves two subtasks: (*i*) *discourse segmentation*, or breaking the text into a sequence of EDUs, and (*ii*) *discourse parsing*, or the task of linking the units (EDUs and larger discourse units) into labeled discourse trees. Recently, Joty et al. (2012) proposed discriminative models for both discourse segmentation and discourse parsing at the sentence level. The segmenter uses a maximum entropy model that achieves state-of-the-art accuracy on this task, having an F_1-score of 90.5%, while human agreement is 98.3%.

The discourse parser uses a dynamic Conditional Random Field (Sutton et al., 2007) as a parsing model in order to infer the probability of all possible discourse tree constituents. The inferred (posterior) probabilities are then used in a probabilistic CKY-like bottom-up parsing algorithm to find the most likely DT. Using the standard set of 18 coarse-grained relations defined in (Carlson and Marcu, 2001), the parser achieved an F_1-score of 79.8%, which is very close to the human agreement of 83%. These high scores allowed us to develop successful discourse similarity metrics.[2]

3.2 Measuring Similarity

A number of metrics have been proposed to measure the similarity between two labeled trees, e.g., Tree Edit Distance (Tai, 1979) and Tree Kernels (Collins and Duffy, 2001; Moschitti and Basili, 2006). Tree kernels (TKs) provide an effective way to integrate arbitrary tree structures in kernel-based machine learning algorithms like SVMs.

In the present work, we use the convolution TK defined in (Collins and Duffy, 2001), which efficiently calculates the number of common subtrees in two trees. Note that this kernel was originally designed for syntactic parsing, where the subtrees are subject to the constraint that their nodes are taken with either all or none of the children. This constraint of the TK imposes some limitations on the type of substructures that can be compared.

[2]The discourse parser is freely available from http://alt.qcri.org/tools/

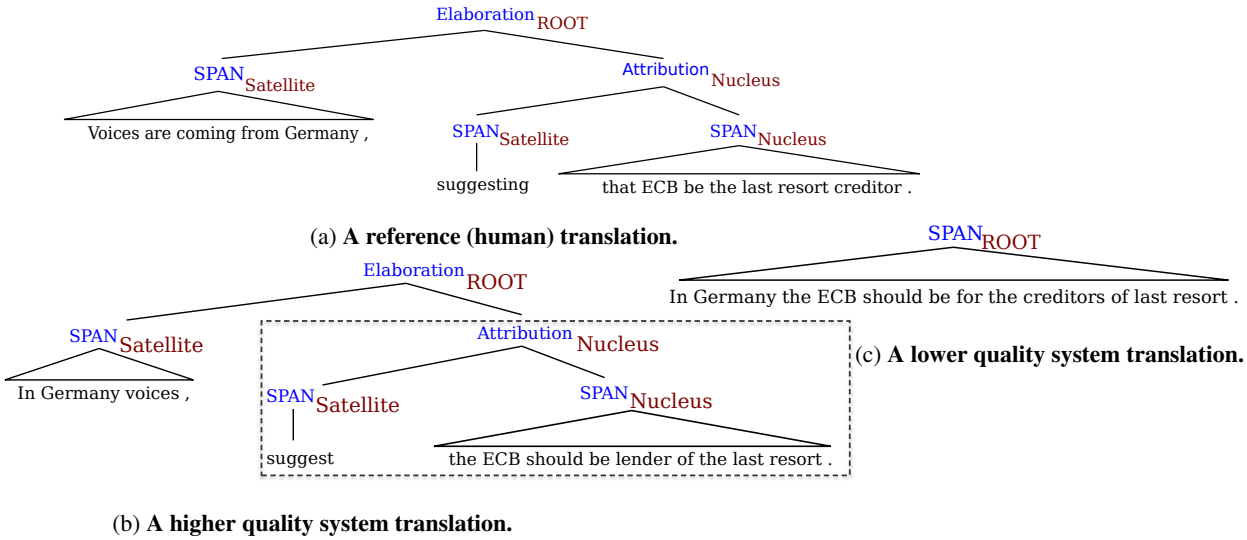


Figure 1: Example of three different discourse trees for the translations of a source sentence. (a) The reference, (b) A higher quality translation, (c) A lower quality translation.

One way to cope with the limitations of the TK is to change the representation of the trees to a form that is suitable to capture the relevant information for our task. We experiment with TKs applied to two different representations of the discourse tree: non-lexicalized (DR), and lexicalized (DR-LEX). In Figure 2 we show the two representations for the subtree that spans the text: *"suggest the ECB should be the lender of last resort"*, which is highlighted in Figure 1b.

As shown in Figure 2a, DR does not include any lexical item, and therefore measures the similarity between two translations in terms of their discourse structures only. On the contrary, DR-LEX includes the lexical items to account for lexical matching; moreover, it separates the structure (the skeleton) of the tree from its labels, i.e. the nuclearity and the relations, in order to allow the tree kernel to give partial credit to subtrees that differ in labels but match in their skeletons. More specifically, it uses the tags SPAN and EDU to build the skeleton of the tree, and considers the nuclearity and/or the relation labels as properties, added as children, of these tags.

For example, a SPAN has two properties (its nuclearity and its relation), and an EDU has one property (its nuclearity). The words of an EDU are placed under the predefined children NGRAM. In order to allow the tree kernel to find subtree matches at the word level, we include an additional layer of *dummy* leaves as was done in (Moschitti et al., 2007); not shown in Figure 2, for simplicity.

4 Experimental Setup

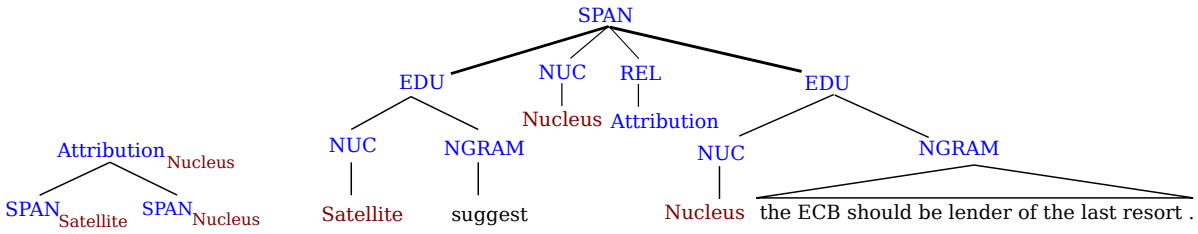
In our experiments, we used the data available for the WMT12 and the WMT11 metrics shared tasks for translations into English.[3] This included the output from the systems that participated in the WMT12 and the WMT11 MT evaluation campaigns, both consisting of 3,003 sentences, for four different language pairs: Czech-English (CS-EN), French-English (FR-EN), German-English (DE-EN), and Spanish-English (ES-EN); as well as a dataset with the English references.

We measured the correlation of the metrics with the human judgments provided by the organizers. The judgments represent rankings of the output of five systems chosen at random, for a particular sentence, also chosen at random. Note that each judgment effectively constitutes 10 pairwise system rankings. The overall coverage, i.e. the number of unique sentences that were evaluated, was only a fraction of the total; the total number of judgments, along with other information of the datasets are shown in Table 1.

4.1 MT Evaluation Metrics

In this study, we evaluate to what extent existing evaluation metrics can benefit from additional discourse information. To do so, we contrast different MT evaluation metrics with and without discourse information. The evaluation metrics we used are described below.

[3] http://www.statmt.org/wmt{11,12}/results.html



(a) DT for DR (b) DT for DR-LEX

Figure 2: Two different DT representations for the highlighted subtree shown in Figure 1b.

	WMT12				WMT11			
	systs	ranks	sents	judges	systs	ranks	sents	judges
CS-EN	6	1,294	951	45	8	498	171	20
DE-EN	16	1,427	975	47	20	924	303	31
ES-EN	12	1,141	923	45	15	570	207	18
FR-EN	15	1,395	949	44	18	708	249	32

Table 1: Number of systems (systs), judgments (ranks), unique sentences (sents), and different judges (judges) for the different language pairs, for the human evaluation of the WMT12 and WMT11 shared tasks.

Metrics from WMT12. We used the publicly available scores for all metrics that participated in the WMT12 metrics task (Callison-Burch et al., 2012): SPEDE07PP, AMBER, METEOR, TERRORCAT, SIMPBLEU, XENERRCATS, WORDBLOCKEC, BLOCKERRCATS, and POSF.

Metrics from ASIYA. We used the freely available version of the ASIYA toolkit[4] in order to extend the set of evaluation measures contrasted in this study beyond those from the WMT12 metrics task. ASIYA (Giménez and Màrquez, 2010a) is a suite for MT evaluation that provides a large set of metrics that use different levels of linguistic information. For reproducibility, below we explain the individual metrics with the exact names required by the toolkit to calculate them.

First, we used ASIYA's ULC (Giménez and Màrquez, 2010b), which was the best performing metric at the system and the segment levels at the WMT08 and WMT09 metrics tasks. This is a uniform linear combination of 12 individual metrics. From the original ULC, we only replaced TER and Meteor individual metrics by newer versions taking into account synonymy lookup and paraphrasing: TERp-A and METEOR-pa in ASIYA's terminology. We will call this combined metric Asiya-0809 in our experiments.

[4]http://nlp.lsi.upc.edu/asiya/

To complement the set of individual metrics that participated at the WMT12 metrics task, we also computed the scores of other commonly-used evaluation metrics: BLEU (Papineni et al., 2002), NIST (Doddington, 2002), TER (Snover et al., 2006), ROUGE-W (Lin, 2004), and three METEOR variants (Denkowski and Lavie, 2011): METEOR-ex (exact match), METEOR-st (+stemming) and METEOR-sy (+synonyms). The uniform linear combination of the previous 7 individual metrics plus the 12 from Asiya-0809 is reported as Asiya-ALL in the experimental section.

The individual metrics combined in Asiya-ALL can be naturally categorized according to the type of linguistic information they use to compute the quality scores. We grouped them in the following four families and calculated the uniform linear combination of the metrics in each group:[5]

1. Asiya-LEX. Combination of five metrics based on lexical similarity: BLEU, NIST, METEOR-ex, ROUGE-W, and TERp-A.
2. Asiya-SYN. Combination of four metrics ba-sed on syntactic information from constituency and dependency parse trees: 'CP-STM-4', 'DP-HWCM_c-4', 'DP-HWCM_r-4', and 'DP-Or(*)'.
3. Asiya-SRL. Combination of three metric variants based on predicate argument structures (semantic role labeling): 'SR-Mr(*)', 'SR-Or(*)', and 'SR-Or'.
4. Asiya-SEM. Combination of two metrics variants based on semantic parsing:[6] 'DR-Or(*)' and 'DR-Orp(*)'.

[5]A detailed description of every individual metric can be found at (Giménez and Màrquez, 2010b). For a more up-to-date description, see the User Manual from ASIYA's website.

[6]In ASIYA the metrics from this family are referred to as "Discourse Representation" metrics. However, the structures they consider are actually very different from the discourse structures exploited in this paper. See the discussion in Section 2. For clarity, we will refer to them as *semantic parsing* metrics.

All uniform linear combinations are calculated outside ASIYA. In order to make the scores of the different metrics comparable, we performed a $\min$–$\max$ normalization, for each metric, and for each language pair combination.

4.2 Human Judgements and Learning

The human-annotated data from the WMT campaigns encompasses series of rankings on the output of different MT systems for every source sentence. Annotators rank the output of five systems according to perceived translation quality. The organizers relied on a random selection of systems, and a large number of comparisons between pairs of them, to make comparisons across systems feasible (Callison-Burch et al., 2012). As a result, for each source sentence, only relative rankings were available. As in the WMT12 experimental setup, we use these rankings to calculate correlation with human judgments at the sentence-level, i.e. Kendall's Tau; see (Callison-Burch et al., 2012) for details.

For the experiments reported in Section 5.4, we used pairwise rankings to discriminatively learn the weights of the linear combinations of individual metrics. In order to use the WMT12 data for training a learning-to-rank model, we transformed the five-way relative rankings into ten pairwise comparisons. For instance, if a judge ranked the output of systems A, B, C, D, E as $A > B > C > D > E$, this would entail that $A > B$, $A > C$, $A > D$ and $A > E$, etc.

To determine the relative weights for the tuned combinations, we followed a similar approach to the one used by PRO to tune the relative weights of the components of a log-linear SMT model (Hopkins and May, 2011), also using Maximum Entropy as the base learning algorithm. Unlike PRO, (*i*) we use *human judgments*, not automatic scores, and (*ii*) we train on *all pairs*, not on a subsample.

5 Experimental Results

In this section, we explore how discourse information can be used to improve machine translation evaluation metrics. Below we present the evaluation results at the system- and segment-level, using our two basic metrics on discourse trees (Section 3.1), which are referred to as DR and DR-LEX.

5.1 Evaluation

In our experiments, we only consider translation into English, and use the data described in Table 1. For evaluation, we follow the setup of the metrics task of WMT12 (Callison-Burch et al., 2012): at the system-level, we use the official script from WMT12 to calculate the Spearman's correlation, where higher *absolute* values indicate better metrics performance; at the segment-level, we use Kendall's Tau for measuring correlation, where negative values are worse than positive ones.[7]

In our experiments, we combine DR and DR-LEX to other metrics in two different ways: using uniform linear interpolation (at system- and segment-level), and using a tuned linear interpolation for the segment-level. We only present the average results over all four language pairs. For simplicity, in our tables we show results divided into evaluation groups:

1. Group I: contains our evaluation metrics, DR and DR-LEX.

2. Group II: includes the metrics that participated in the WMT12 metrics task, excluding metrics which did not have results for all language pairs.

3. Group III: contains other important evaluation metrics, which were not considered in the WMT12 metrics task: NIST and ROUGE for both system- and segment-level, and BLEU and TER at segment-level.

4. Group IV: includes the metric combinations calculated with ASIYA and described in Section 4.

For each metric in groups II, III and IV, we present the results for the original metric as well for the linear interpolation of that metric with DR and with DR-LEX. The combinations with DR and DR-LEX that improve over the original metrics are shown in **bold**, and those that degrade are in *italic*. Furthermore, we also present overall results for: (*i*) the average score over all metrics, excluding DR and DR-LEX, and (*ii*) the differences in the correlations for the DR/DR-LEX-combined and the original metrics.

[7] We have fixed a bug in the scoring tool from WMT12, which was making all scores positive. This made TERRORCAT's score negative, as we present it in Table 3.

	Metrics		+DR	+DR-LEX
I	DR	.807	–	–
	DR-LEX	.876	–	–
II	SEMPOS	.902	*.853*	**.903**
	AMBER	.857	*.829*	**.869**
	METEOR	.834	**.861**	**.888**
	TERRORCAT	.831	**.854**	**.889**
	SIMPBLEU	.823	**.826**	**.859**
	TER	.812	**.836**	**.848**
	BLEU	.810	**.830**	**.846**
	POSF	.754	**.841**	**.857**
	BLOCKERRCATS	.751	**.859**	**.855**
	WORDBLOCKEC	.738	**.822**	**.843**
	XENERRCATS	.735	**.819**	**.843**
III	NIST	.817	**.842**	**.875**
	ROUGE	.884	**.899**	*.869*
IV	Asiya-LEX	.879	**.881**	**.882**
	Asiya-SYN	.891	**.913**	*.883*
	Asiya-SRL	.917	*.911*	*.909*
	Asiya-SEM	.891	*.889*	*.886*
	Asiya-0809	.905	**.914**	.905
	Asiya-ALL	.899	**.907**	*.896*
	average	.839	**.862**	**.874**
	diff.		**+.024**	**+.035**

Table 2: Results on WMT12 at the system-level. Spearman's correlation with human judgments.

	Metrics		+DR	+DR-LEX
I	DR	-.433	–	–
	DR-LEX	.133	–	–
II	SPEDE07PP	.254	*.190*	*.223*
	METEOR	.247	*.178*	*.217*
	AMBER	.229	*.180*	*.216*
	SIMPBLEU	.172	*.141*	**.191**
	XENERRCATS	.165	*.132*	**.185**
	POSF	.154	*.125*	**.201**
	WORDBLOCKEC	.153	*.122*	**.181**
	BLOCKERRCATS	.074	*.068*	**.151**
	TERRORCAT	-.186	**-.111**	**-.104**
III	NIST	.214	*.172*	*.206*
	ROUGE	.185	*.144*	**.201**
	TER	.217	*.179*	**.229**
	BLEU	.185	*.154*	**.190**
IV	Asiya-LEX	.254	*.237*	*.253*
	Asiya-SYN	.177	*.169*	**.191**
	Asiya-SRL	-.023	**.015**	**.161**
	Asiya-SEM	.134	**.152**	**.197**
	Asiya-0809	.254	*.250*	**.258**
	Asiya-ALL	.268	*.265*	**.270**
	average	.165	*.145*	**.190**
	diff.		*-.019*	**+.026**

Table 3: Results on WMT12 at the segment-level. Kendall's Tau with human judgments.

5.2 System-level Results

Table 2 shows the system-level experimental results for WMT12. We can see that DR is already competitive by itself: on average, it has a correlation of .807, very close to BLEU and TER scores (.810 and .812, respectively). Moreover, DR yields improvements when combined with 15 of the 19 metrics; worsening only four of the metrics. Overall, we observe an average improvement of +.024, in the correlation with the human judgments. This suggests that DR contains information that is complementary to that used by the other metrics. Note that this is true both for the individual metrics from groups II and III, as well as for the metric combinations in group IV. Combinations in the last group involve several metrics that already use linguistic information at different levels and are hard to improve over; yet, adding DR does improve, which shows that it has some complementary information to offer.

As expected, DR-LEX performs better than DR since it is lexicalized (at the unigram level), and also gives partial credit to correct structures. Individually, DR-LEX outperforms most of the metrics from group II, and ranks as the second best metric in that group. Furthermore, when combined with individual metrics in group II, DR-LEX is able to improve consistently over each one of them.

Note that, even though DR-LEX has better individual performance than DR, it does not yield improvements when combined with most of the metrics in group IV.[8] However, over all metrics and all language pairs, DR-LEX is able to obtain an average improvement in correlation of +.035, which is remarkably higher than that of DR. Thus, we can conclude that at the system-level, adding discourse information to a metric, even using the simplest of the combination schemes, is a good idea for most of the metrics, and can help to significantly improve the correlation with human judgments.

5.3 Segment-level Results: Non-tuned

Table 3 shows the results for WMT12 at the segment-level. We can see that DR performs badly, with a high negative Kendall's Tau of -.433. This should not be surprising: (a) the discourse tree structure alone does not contain enough information for a good evaluation at the segment-level, and (b) this metric is more sensitive to the quality of the DT, which can be wrong or void.

[8] In this work, we have not investigated the reasons behind this phenomenon. We speculate that this might be caused by the fact that the lexical information in DR-LEX is incorporated only in the form of unigram matching at the sentence-level, while the metrics in group IV are already complex combined metrics, which take into account stronger lexical models. Note, however, that the variations are very small and might not be significant.

	Metrics	Orig.	Tuned	+DR	+DR-LEX
I	DR	-.433	–	–	–
	DR-LEX	.133	–	–	–
II	SPEDE07PP	.254	–	*.253*	.254
	METEOR	.247	–	**.250**	**.251**
	AMBER	.229	–	**.230**	**.232**
	SIMPBLEU	.172	–	**.181**	**.199**
	TERRORCAT	-.186	–	**.181**	**.196**
	XENERRCATS	.165	–	**.175**	**.194**
	POSF	.154	–	**.160**	**.201**
	WORDBLOCKEC	.153	–	**.161**	**.189**
	BLOCKERRCATS	.074	–	**.087**	**.150**
III	NIST	.214	–	**.222**	**.224**
	ROUGE	.185	–	**.196**	**.218**
	TER	.217	–	**.229**	**.246**
	BLEU	.185	–	**.189**	**.194**
IV	Asiya-LEX	.254	.266	**.269**	**.270**
	Asiya-SYN	.177	.229	.228	**.232**
	Asiya-SRL	-.023	-.004	**.039**	**.181**
	Asiya-SEM	.134	.146	**.179**	**.202**
	Asiya-0809	.254	.295	.295	.295
	Asiya-ALL	.268	.296	.295	.295
	average	.165		.201	.222
	diff.			**+.036**	**+.057**

Table 4: Results on WMT12 at the segment-level: tuning with cross-validation on WMT12. Kendall's Tau with human judgments.

Additionally, DR is more likely to produce a high number of ties, which is harshly penalized by WMT12's definition of Kendall's Tau. Conversely, ties and incomplete discourse analysis were not a problem at the system-level, where evidence from all 3,003 test sentences is aggregated, and allows to rank systems more precisely. Due to the low score of DR as an individual metric, it fails to yield improvements when uniformly combined with other metrics.

Again, DR-LEX is better than DR; with a positive Tau of +.133, yet as an individual metric, it ranks poorly compared to other metrics in group II. However, when linearly combined with other metrics, DR-LEX outperforms 14 of the 19 metrics in Table 3. Across all metrics, DR-LEX yields an average Tau improvement of +.026, i.e. from .165 to .190. This is a large improvement, taking into account that the combinations are just uniform linear combinations. In subsection 5.4, we present the results of tuning the linear combination in a discriminative way.

5.4 Segment-level Results: Tuned

We experimented with tuning the weights of the individual metrics in the metric combinations, using the learning method described in Section 4.2. First, we did this using cross-validation to tune and test on WMT12. Later we tuned on WMT12 and evaluated on WMT11. For cross-validation in WMT12, we used ten folds of approximately equal sizes, each containing about 300 sentences: we constructed the folds by putting together entire documents, thus not allowing sentences from a document to be split over two different folds. During each cross-validation run, we trained our pairwise ranker using the human judgments corresponding to nine of the ten folds. We aggregated the data for different language pairs, and produced a single set of tuning weights for all language pairs.[9] We then used the remaining fold for evaluation

The results are shown in Table 4. As in previous sections we present the average results over all four language pairs. We can see that the tuned combinations with DR-LEX improve over most of the individual metrics in groups II and III. Interestingly, the tuned combinations that include the much weaker metric DR now improve over 12 out of 13 of the individual metrics in groups II and III, and only slightly degrades the score of the 13th one (SPEDE07PP).

Note that the ASIYA metrics are combinations of several metrics, and these combinations (which exclude DR and DR-LEX) can be also tuned; this yields sizable improvements over the untuned versions as column three in the table shows. Compared to this baseline, DR improves for three of the six ASIYA metrics, while DR-LEX improves for four of them. Note that improving over the last two ASIYA metrics is very hard: they have very high scores of .296 and .295; for comparison, the best segment-level system at WMT12 (SPEDE07PP) achieved a Tau of .254.

On average, DR improves Tau from .165 to .201, which is +.036, while DR-LEX improves to .222, or +.057. These much larger improvements highlight the importance of tuning the linear combination when working at the segment-level.

5.4.1 Testing on WMT11

In order to rule out the possibility that the improvement of the tuned metrics on WMT12 comes from over-fitting, and to verify that the tuned metrics do generalize when applied to other sentences, we also tested on a new test set: WMT11.

[9] Tuning separately for each language pair yielded slightly lower results.

Therefore, we tuned the weights on *all* WMT12 pairwise judgments (no cross-validation), and we evaluated on WMT11. Since the metrics that participated in WMT11 and WMT12 are different (and even when they have the same name, there is no guarantee that they have not changed from 2011 to 2012), we only report results for the versions of NIST, ROUGE, TER, and BLEU available in ASIYA, as well as for the ASIYA metrics, thus ensuring that the metrics in the experiments are consistent for 2011 and 2012.

The results are shown in Table 5. Once again, tuning yields sizable improvements over the simple combination for the ASIYA metrics (third column in Table 5). Adding DR and DR-LEX to the combinations manages to improve over five and four of the six tuned ASIYA metrics, respectively. However, some of the differences are very small. On the contrary, DR and DR-LEX significantly improve over NIST, ROUGE, TER, and BLEU. Overall, DR improves the average Tau from .207 to .244, which is +.037, while DR-LEX improves to .267 or +.061. These improvements are very close to those for the WMT12 cross-validation. This shows that the weights learned on WMT12 generalize well, as they are also good for WMT11.

What is also interesting to note is that when tuning is used, DR helps achieve sizeable improvements, even if not as strong as for DR-LEX. This is remarkable given that DR has a strong negative Tau as an individual metric at the sentence-level. This suggests that both DR and DR-LEX contain information that is complementary to that of the individual metrics that we experimented with.

Overall, from the experimental results in this section, we can conclude that discourse structure is an important information source to be taken into account in the automatic evaluation of machine translation output.

			Tuned		
	Metrics	**Orig.**		+DR	+DR-LEX
I	DR	-.447	–	–	–
	DR-LEX	.146	–	–	–
III	NIST	.219	–	**.226**	**.232**
	ROUGE	.205	–	**.218**	**.242**
	TER	.262	–	**.274**	**.296**
	BLEU	.186	–	**.192**	**.207**
IV	Asiya-LEX	.282	.301	**.302**	**.303**
	Asiya-SYN	.216	.259	**.260**	**.260**
	Asiya-SRL	-.004	.017	**.051**	**.200**
	Asiya-SEM	.189	.194	**.220**	**.239**
	Asiya-0809	.300	.348	**.349**	.348
	Asiya-ALL	.313	.347	.347	.347
	average	.207		.244	.267
	diff.			**+.037**	**+.061**

Table 5: Results on WMT11 at the segment-level: tuning on the entire WMT12. Kendall's Tau with human judgments.

6 Conclusions and Future Work

In this paper we have shown that discourse structure can be used to improve automatic MT evaluation. First, we defined two simple discourse-aware similarity metrics (lexicalized and un-lexicalized), which use the all-subtree kernel to compute similarity between discourse parse trees in accordance with the Rhetorical Structure Theory. Then, after extensive experimentation on WMT12 and WMT11 data, we showed that a variety of existing evaluation metrics can benefit from our discourse-based metrics, both at the segment- and the system-level, especially when the discourse information is incorporated in an informed way (i.e. using supervised tuning). Our results show that discourse-based metrics can improve the state-of-the-art MT metrics, by increasing correlation with human judgments, even when only sentence-level discourse information is used.

Addressing discourse-level phenomena in MT is a relatively new research direction. Yet, many of the ongoing efforts have been moderately successful according to traditional evaluation metrics. There is a consensus in the MT community that more discourse-aware metrics need to be proposed for this area to move forward. We believe this work is a valuable contribution towards this longer-term goal.

The tuned combined metrics tested in this paper are just an initial proposal, i.e. a simple adjustment of the relative weights for the individual metrics in a linear combination. In the future, we plan to work on integrated representations of syntactic, semantic and discourse-based structures, which would allow us to train evaluation metrics based on more fine-grained features. Additionally, we propose to use the discourse information for MT in two different ways. First, at the sentence-level, we can use discourse information to re-rank alternative MT hypotheses; this could be applied either for MT parameter tuning, or as a post-processing step for the MT output. Second, we propose to move in the direction of using discourse information beyond the sentence-level.

References

Nicholas Asher and Alex Lascarides, 2003. *Logics of Conversation*. Cambridge University Press.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Omar Zaidan. 2011. Findings of the 2011 workshop on statistical machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 22–64, Edinburgh, Scotland, July. ACL.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 workshop on statistical machine translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada, June. ACL.

Lynn Carlson and Daniel Marcu. 2001. Discourse Tagging Reference Manual. Technical Report ISI-TR-545, University of Southern California Information Sciences Institute.

Bruno Cartoni, Sandrine Zufferey, Thomas Meyer, and Andrei Popescu-Belis. 2011. How comparable are parallel corpora? measuring the distribution of general vocabulary and connectives. In *Proceedings of the 4th Workshop on Building and Using Comparable Corpora: Comparable Corpora and the Web*, pages 78–86, Portland, Oregon, June. ACL.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP'08)*, Honolulu, Hawaii, USA.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 263–270, Ann Arbor, Michigan.

Michael Collins and Nigel Duffy. 2001. Convolution Kernels for Natural Language. In *Neural Information Processing Systems*, NIPS'01, pages 625–632, Vancouver, Canada.

Elisabet Comelles, Jesús Giménez, Lluís Màrquez, Irene Castellón, and Victoria Arranz. 2010. Document-level automatic mt evaluation based on discourse representations. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 333–338, Uppsala, Sweden, July. ACL.

Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 85–91, Edinburgh, Scotland, July. ACL.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*, HLT '02, pages 138–145, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of the 2004 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technology*, HLT-NAACL, pages 273–280.

Jesús Giménez and Lluís Màrquez. 2007. Linguistic features for automatic evaluation of heterogenous MT systems. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 256–264, Prague, Czech Republic, June. ACL.

Jesús Giménez and Lluís Màrquez. 2009. On the robustness of syntactic and semantic features for automatic MT evaluation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 250–258, Athens, Greece, March. ACL.

Jesús Giménez and Lluís Màrquez. 2010a. Asiya: an Open Toolkit for Automatic Machine Translation (Meta-)Evaluation. *The Prague Bulletin of Mathematical Linguistics*, 94:77–86.

Jesús Giménez and Lluís Màrquez. 2010b. Linguistic Measures for Automatic Machine Translation Evaluation. *Machine Translation*, 24(3–4):77–86.

Michael Halliday and Ruqaiya Hasan, 1976. *Cohesion in English*. Longman, London.

Christian Hardmeier and Marcello Federico. 2010. Modelling pronominal anaphora in statistical machine translation. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 283–289.

Christian Hardmeier, Joakim Nivre, and Jörg Tiedemann. 2012. Document-wide decoding for phrase-based statistical machine translation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1179–1190, Jeju Island, Korea. ACL.

Christian Hardmeier. 2012. Discourse in statistical machine translation. a survey and a case study. *Discours. Revue de linguistique, psycholinguistique et informatique*, 11(8726).

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, EMNLP '11.

Shafiq Joty, Giuseppe Carenini, and Raymond T. Ng. 2012. A Novel Discriminative Framework for Sentence-Level Discourse Analysis. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 904–915, Jeju Island, Korea. ACL.

Shafiq Joty, Giuseppe Carenini, Raymond T. Ng, and Yashar Mehdad. 2013. Combining Intra- and Multi-sentential Rhetorical Parsing for Document-level Discourse Analysis. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, ACL '13, pages 486–496, Sofia, Bulgaria. ACL.

Hans Kamp and Uwe Reyle. 1993. *From Discourse to Logic: Introduction to Model theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory.* Number 42 in Studies in Linguistics and Philosophy. Kluwer Academic Publishers.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Proceedings of Workshop on Text Summarization Branches Out*, pages 74–81, Barcelona.

Ding Liu and Daniel Gildea. 2005. Syntactic features for evaluation of machine translation. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 25–32, Ann Arbor, Michigan, June. ACL.

Chi-kiu Lo, Anand Karthik Tumuluru, and Dekai Wu. 2012. Fully automatic semantic mt evaluation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 243–252, Montréal, Canada, June. ACL.

William Mann and Sandra Thompson. 1988. Rhetorical Structure Theory: Toward a Functional Theory of Text Organization. *Text*, 8(3):243–281.

Thomas Meyer, Andrei Popescu-Belis, Najeh Hajlaoui, and Andrea Gesmundo. 2012. Machine translation of labeled discourse connectives. In *Proceedings of the Tenth Biennial Conference of the Association for Machine Translation in the Americas (AMTA)*.

Thomas Meyer. 2011. Disambiguating temporal-contrastive connectives for machine translation. In *Proceedings of the ACL 2011 Student Session*, pages 46–51, Portland, OR, USA, June. ACL.

Alessandro Moschitti and Roberto Basili. 2006. A Tree Kernel approach to Question and Answer Classification in Question Answering Systems. In *Proceedings of the 5th international conference on Language Resources and Evaluation*, Genoa, Italy.

Alessandro Moschitti, Silvia Quarteroni, Roberto Basili, and Suresh Manandhar. 2007. Exploiting Syntactic and Shallow Semantic Kernels for Question/Answer Classification. In *Proceedings of the ACL-2007*, pages 776–783, Prague, Czech Republic.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the Association for Computational Linguistics (ACL'02)*, Philadelphia, PA, USA.

Maja Popovic and Hermann Ney. 2007. Word error rates: Decomposition over POS classes and applications for error analysis. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 48–55, Prague, Czech Republic, June. ACL.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal smt. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 271–279, Ann Arbor, Michigan.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Biennial Conference of the Association for Machine Translation in the Americas*, AMTA '06, Cambridge, MA, USA.

Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. 2007. Dynamic Conditional Random Fields: Factorized Probabilistic Models for Labeling and Segmenting Sequence Data. *Journal of Machine Learning Research (JMLR)*, 8:693–723.

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

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '07, Prague, Czech Republic.

Bonnie Webber, Andrei Popescu-Belis, Katja Markert, and Jörg Tiedemann, editors. 2013. *Proceedings of the Workshop on Discourse in Machine Translation*. ACL, Sofia, Bulgaria, August.

Bonnie Webber. 2004. D-LTAG: Extending Lexicalized TAG to Discourse. *Cognitive Science*, 28(5):751–779.

Billy T. M. Wong and Chunyu Kit. 2012. Extending machine translation evaluation metrics with lexical cohesion to document level. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL, pages 1060–1068, Jeju Island, Korea, July. ACL.

Dekai Wu and Pascale Fung. 2009. Semantic roles for smt: A hybrid two-pass model. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 13–16, Boulder, Colorado, June.

Learning Continuous Phrase Representations for Translation Modeling

Jianfeng Gao Xiaodong He Wen-tau Yih Li Deng
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
{jfgao,xiaohe,scottyih,deng}@microsoft.com

Abstract

This paper tackles the sparsity problem in estimating phrase translation probabilities by learning continuous phrase representations, whose distributed nature enables the sharing of related phrases in their representations. A pair of source and target phrases are projected into continuous-valued vector representations in a low-dimensional latent space, where their translation score is computed by the distance between the pair in this new space. The projection is performed by a neural network whose weights are learned on parallel training data. Experimental evaluation has been performed on two WMT translation tasks. Our best result improves the performance of a state-of-the-art phrase-based statistical machine translation system trained on WMT 2012 French-English data by up to 1.3 BLEU points.

1 Introduction

The phrase translation model, also known as the *phrase table*, is one of the core components of phrase-based statistical machine translation (SMT) systems. The most common method of constructing the phrase table takes a two-phase approach (Koehn et al. 2003). First, the bilingual phrase pairs are extracted heuristically from an automatically word-aligned training data. The second phase, which is the focus of this paper, is parameter estimation where each phrase pair is assigned with some scores that are estimated based on counting these phrases or their words using the same word-aligned training data.

Phrase-based SMT systems have achieved state-of-the-art performance largely due to the fact that long phrases, rather than single words, are used as translation units so that useful context information can be captured in selecting translations. However, longer phrases occur less often in training data, leading to a severe data sparseness problem in parameter estimation. There has been a plethora of research reported in the literature on improving parameter estimation for the phrase translation model (e.g., DeNero et al. 2006; Wuebker et al. 2010; He and Deng 2012; Gao and He 2013).

This paper revisits the problem of scoring a phrase translation pair by developing a Continuous-space Phrase Translation Model (CPTM). The translation score of a phrase pair in this model is computed as follows. First, we represent each phrase as a bag-of-words vector, called *word vector* henceforth. We then project the word vector, in either the source language or the target language, into a respective continuous feature vector in a common low-dimensional space that is language independent. The projection is performed by a multi-layer neural network. The projected feature vector forms the *continuous representation* of a phrase. Finally, the translation score of a source-target phrase pair is computed by the distance between their feature vectors.

The main motivation behind the CPTM is to alleviate the data sparseness problem associated with the traditional counting-based methods by grouping phrases with a similar meaning across different languages. This style of grouping is made possible because of the distributed nature of the continuous-space representations for phrases. No such sharing was possible in the original symbolic space for representing words or phrases. In this model, semantically or grammatically related phrases, in both the source and the target languages, would tend to have similar (close) feature vectors in the continuous space, guided by the training objective. Since the translation score is a smooth function of these feature vectors, a small

change in the features should only lead to a small change in the translation score.

The primary research task in developing the CPTM is learning the continuous representation of a phrase that is effective for SMT. Motivated by recent studies on continuous-space language models (e.g., Bengio et al. 2003; Mikolov et al. 2011; Schwenk et al., 2012), we use a neural network to project a word vector to a feature vector. Ideally, the projection would discover those latent features that are useful to differentiate *good* translations from *bad* ones, for a given source phrase. However, there is no training data with explicit annotation on the quality of phrase translations. The phrase translation pairs are *hidden* in the parallel source-target sentence pairs, which are used to train the traditional translation models. The quality of a phrase translation can only be judged implicitly through the translation quality of the sentences, as measured by BLEU, which contain the phrase pair. In order to overcome this challenge and let the BLEU metric guide the projection learning, we propose a new method to learn the parameters of a neural network. This new method, via the choice of an appropriate objective function in training, automatically forces the feature vector of a source phrase to be closer to the feature vectors of its candidate translations. As a result, the BLEU score is improved when these translations are selected by an SMT decoder to produce final, sentence-level translations. The new learning method makes use of the L-BFGS algorithm and the expected BLEU as the objective function defined on N-best lists.

To the best of our knowledge, the CPTM proposed in this paper is the first continuous-space phrase translation model that makes use of joint representations of a phrase in the source language and its translation in the target language (to be detailed in Section 4) and that is shown to lead to significant improvement over a standard phrase-based SMT system (to be detailed in Section 6).

Like the traditional phrase translation model, the translation score of each bilingual phrase pair is modeled explicitly in our model. However, instead of estimating the phrase translation score on aligned parallel data, our model intends to capture the grammatical and semantic similarity between a source phrase and its paired target phrase by projecting them into a common, continuous space that is language independent.

The rest of the paper is organized as follows. Section 2 reviews previous work. Section 3 reviews the log-linear model for phrase-based SMT and Sections 4 presents the CPTM. Section 5 describes the way the model parameters are estimated, followed by the experimental results in Section 6. Finally, Section 7 concludes the paper.

2 Related Work

Representations of words or documents as continuous vectors have a long history. Most of the earlier latent semantic models for learning such vectors are designed for information retrieval (Deerwester et al. 1990; Hofmann 1999; Blei et al. 2003). In contrast, recent work on continuous space language models, which estimate the probability of a word sequence in a continuous space (Bengio et al. 2003; Mikolov et al. 2010), have advanced the state of the art in language modeling, outperforming the traditional n-gram model on speech recognition (Mikolov et al. 2012; Sundermeyer et al. 2013) and machine translation (Mikolov 2012; Auli et al. 2013).

Because these models are developed for monolingual settings, word embedding from these models is not directly applicable to translation. As a result, variants of such models for cross-lingual scenarios have been proposed so that words in different languages are projected into the shared latent vector space (Dumais et al. 1997; Platt et al. 2010; Vinokourov et al. 2002; Yih et al. 2011; Gao et al. 2011; Huang et al. 2013; Zou et al. 2013). In principle, a phrase table can be derived using any of these cross-lingual models, although decoupling the derivation from the SMT training often results in suboptimal performance (e.g., measured in BLEU), as we will show in Section 6.

Recently, there is growing interest in applying continuous-space models for translation. The most related to this study is the work of continuous space n-gram translation models (Schwenk et al. 2007; Schwenk 2012; Son et al. 2012), where the feed-forward neural network language model is extended to represent translation probabilities. However, these earlier studies focused on the *n-gram translation models*, where the translation probability of a phrase or a sentence is decomposed as a product of n-gram probabilities as in a standard n-gram language model. Therefore, it is not clear how their approaches can be applied to the phrase translation model[1], which is much more

[1] Niehues et al. (2011) use different translation units in order to integrate the n-gram translation model into the phrase-based approach. However, it is not clear how a continuous version of such a model can be trained efficiently because the factor models used by Son et al. cannot be applied directly.

widely used in modern SMT systems. In contrast, our model learns jointly the representations of a phrase in the source language as well as its translation in the target language. The recurrent continuous translation models proposed by Kalchbrenner and Blunsom (2013) also adopt the recurrent language model (Mikolov et al. 2010). But unlike the n-gram translation models above, they make no Markov assumptions about the dependency of the words in the target sentence. Continuous space models have also been used for generating translations for new words (Mikolov et al. 2013a) and ITG reordering (Li et al. 2013).

There has been a lot of research on improving the phrase table in phrase-based SMT (Marcu and Wong 2002; Lamber and Banchs 2005; Denero et al. 2006; Wuebker et al. 2010; Zhang et al., 2011; He and Deng 2012; Gao and He 2013). Among them, (Gao and He 2013) is most relevant to the work described in this paper. They estimate phrase translation probabilities using a discriminative training method under the N-best reranking framework of SMT. In this study we use the same objective function to learn the continuous representations of phrases, integrating the strengths associated with these earlier studies.

3 The Log-Linear Model for SMT

Phrase-based SMT is based on a log-linear model which requires learning a mapping between input $F \in \mathcal{F}$ to output $E \in \mathcal{E}$. We are given

- Training samples (F_i, E_i) for $i = 1 \ldots N$, where each source sentence F_i is paired with a reference translation in target language E_i;
- A procedure GEN to generate a list of N-best candidates $\mathrm{GEN}(F_i)$ for an input F_i, where GEN in this study is the baseline phrase-based SMT system, i.e., an in-house implementation of the Moses system (Koehn et al. 2007) that does not use the CPTM, and each $E \in \mathrm{GEN}(F_i)$ is labeled by the sentence-level BLEU score (He and Deng 2012), denoted by $\mathrm{sBleu}(E_i, E)$, which measures the quality of E with respect to its reference translation E_i;
- A vector of features $\mathbf{h} \in \mathbb{R}^M$ that maps each (F_i, E) to a vector of feature values[2]; and
- A parameter vector $\boldsymbol{\lambda} \in \mathbb{R}^M$, which assigns a real-valued weight to each feature.

The components $\mathrm{GEN}(.)$, $\mathbf{h}$ and $\boldsymbol{\lambda}$ define a log-linear model that maps F_i to an output sentence as follows:

$$E^* = \underset{(E,A)\in \mathrm{GEN}(F_i)}{\operatorname{argmax}} \boldsymbol{\lambda}^{\mathrm{T}} \mathbf{h}(F_i, E, A) \qquad (1)$$

which states that given $\boldsymbol{\lambda}$ and $\mathbf{h}$, argmax returns the highest scoring translation E^*, maximizing over correspondences A. In phrase-based SMT, A consists of a segmentation of the source and target sentences into phrases and an alignment between source and target phrases. Since computing the argmax exactly is intractable, it is commonly performed approximatedly by beam search (Och and Ney 2004). Following Liang et al. (2006), we assume that every translation candidate is always coupled with a corresponding A, called the *Viterbi derivation*, generated by (1).

4 A Continuous-Space Phrase Translation Model (CPTM)

The architecture of the CPTM is shown in Figures 1 and 2, where for each pair of source and target phrases (f_i, e_j) in a source-target sentence pair, we first project them into feature vectors $\mathbf{y}_{f_i}$ and $\mathbf{y}_{e_j}$ in a latent, continuous space via a neural network with one hidden layer (as shown in Figure 2), and then compute the translation score, $\mathrm{score}(f_i, e_j)$, by the distance of their feature vectors in that space.

We start with a bag-of-words representation of a phrase $\mathbf{x} \in \mathbb{R}^d$, where $\mathbf{x}$ is a word vector and d is the size of the vocabulary consisting of words in both source and target languages, which is set to 200K in our experiments. We then learn to project $\mathbf{x}$ to a low-dimensional continuous space $\mathbb{R}^k$:

$$\phi(\mathbf{x}): \mathbb{R}^d \rightarrow \mathbb{R}^k$$

The projection is performed using a fully connected neural network with one hidden layer and tanh activation functions. Let $\mathbf{W}_1$ be the projection matrix from the input layer to the hidden layer and $\mathbf{W}_2$ the projection matrix from the hidden layer to the output layer, we have

$$\mathbf{y} \equiv \phi(\mathbf{x}) = \tanh\left(\mathbf{W}_2{}^{\mathrm{T}}\left(\tanh\left(\mathbf{W}_1{}^{\mathrm{T}}\mathbf{x}\right)\right)\right) \qquad (2)$$

[2] Our baseline system uses a set of standard features suggested in Koehn et al. (2007), which is also detailed in Section 6.

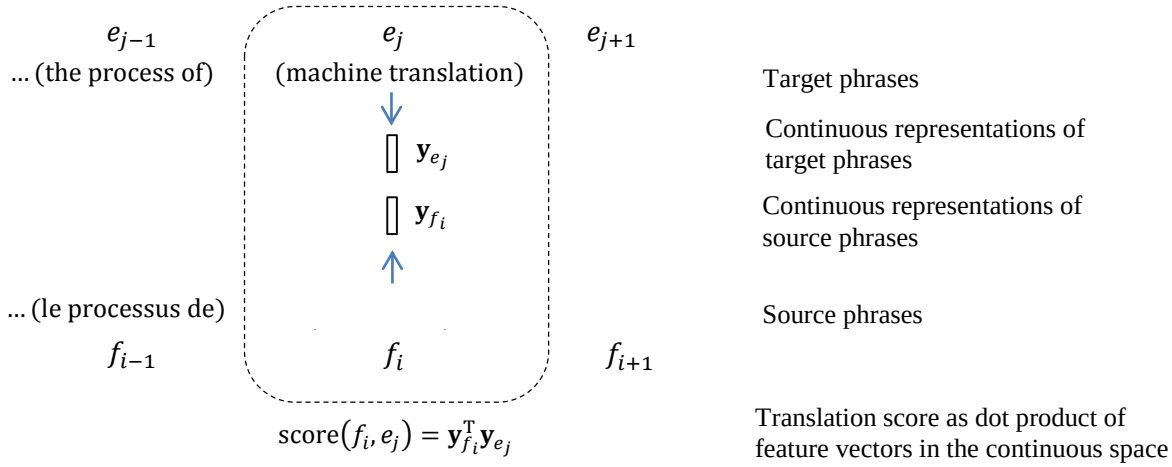


Figure 1. The architecture of the CPTM, where the mapping from a phrase to its continuous representation is shown in Figure 2.

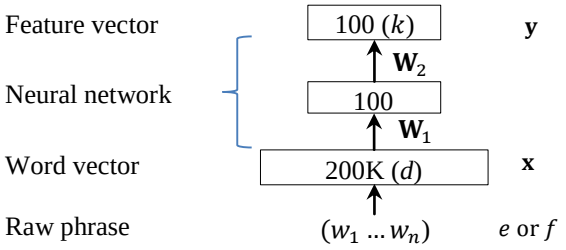


Figure 2. A neural network model for phrases giving rise to their continuous representations. The model with the same form is used for both source and target languages.

The translation score of a source phrase f and a target phrase e can be measured as the similarity (or distance) between their feature vectors. We choose the dot product as the similarity function[3]:

$$\text{score}(f, e) \equiv \text{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e) = \mathbf{y}_f^{\mathrm{T}} \mathbf{y}_e \qquad (3)$$

According to (2), we see that the value of the scoring function is determined by the projection matrices $\boldsymbol{\theta} = \{\mathbf{W}_1, \mathbf{W}_2\}$.

The CPTM of (2) and (3) can be incorporated into the log-linear model for SMT (1) by introducing a new feature h_{M+1} and a new feature weight λ_{M+1}. The new feature is defined as

$$h_{M+1}(F_i, E, A) = \sum_{(f,e)\in A} \text{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e) \qquad (4)$$

Thus, the phrase-based SMT system, into which the CPTM is incorporated, is parameterized by $(\boldsymbol{\lambda}, \boldsymbol{\theta})$, where $\boldsymbol{\lambda}$ is a vector of a handful of parameters used in the log-linear model of (1), with one weight for each feature; and $\boldsymbol{\theta}$ is the projection matrices used in the CPTM defined by (2) and (3). In our experiments we take three steps to learn $(\boldsymbol{\lambda}, \boldsymbol{\theta})$:

1. We use a baseline phrase-based SMT system to generate for each source sentence in training data an N-best list of translation hypotheses[4].
2. We set $\boldsymbol{\lambda}$ to that of the baseline system and let $\lambda_{M+1} = 1$, and optimize $\boldsymbol{\theta}$ w.r.t. a loss function on training data[5].
3. We fix $\boldsymbol{\theta}$, and optimize $\boldsymbol{\lambda}$ using MERT (Och 2003) to maximize BLEU on dev data.

In the next section, we will describe Step 2 in detail as it is directly related to the CPTM training.

[3] In our experiments, we compare dot product and the cosine similarity functions and find that the former works better for nonlinear multi-layer neural networks, and the latter works better for linear neural networks. For the sake of clarity, we choose dot product when we describe the CPTM and its training in Sections 4 and 5, respectively.

[4] The baseline SMT needs to be reasonably good in the sense that the oracle BLEU score on the generated n-best lists needs to be significantly higher than that of the top-1 translations so that the CPTM can be effectively trained.

[5] The initial value of λ_{M+1} can also be tuned using the dev set. However, we find in a pilot study that it is good enough to set it to 1 when the values of all the baseline feature weights, used in the log-linear model of (1), are properly normalized, such as by setting $\lambda_m = \lambda_m / C$ for $m = 1 \dots M$, where C is the unnormalized weight value of the target language model.

5 Training CPTM

This section describes the loss function we employ with the CPTM and the algorithm to train the neural network weights.

We define the loss function $\mathcal{L}(\boldsymbol{\theta})$ as the negative of the N-best list based expected BLEU, denoted by $\text{xBleu}(\boldsymbol{\theta})$. In the reranking framework of SMT outlined in Section 3, $\text{xBleu}(\boldsymbol{\theta})$ over one training sample (F_i, E_i) is defined as

$$\text{xBleu}(\boldsymbol{\theta}) = \sum_{E \in \text{GEN}(F_i)} P(E|F_i)\text{sBleu}(E_i, E) \quad (5)$$

where $\text{sBleu}(E_i, E)$ is the sentence-level BLEU score, and $P(E|F_i)$ is the translation probability from F_i to E computed using *softmax* as

$$P(E|F_i) = \frac{\exp\left(\gamma \boldsymbol{\lambda}^{\mathrm{T}} \mathbf{h}(F_i, E, A)\right)}{\sum_{E' \in \text{GEN}(F_i)} \exp\left(\gamma \boldsymbol{\lambda}^{\mathrm{T}} \mathbf{h}(F_i, E', A)\right)} \quad (6)$$

where $\boldsymbol{\lambda}^{\mathrm{T}}\mathbf{h}$ is the log-linear model of (1), which also includes the feature derived from the CPTM as defined by (4), and γ is a tuned smoothing factor.

Let $\mathcal{L}(\boldsymbol{\theta})$ be a loss function which is differentiable w.r.t. the parameters of the CPTM, $\boldsymbol{\theta}$. We can compute the gradient of the loss and learn $\boldsymbol{\theta}$ using gradient-based numerical optimization algorithms, such as L-BFGS or stochastic gradient descent (SGD).

5.1 Computing the Gradient

Since the loss does not explicitly depend on $\boldsymbol{\theta}$, we use the chain rule for differentiation:

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \sum_{(f,e)} \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \text{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e)} \frac{\partial \text{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e)}{\partial \boldsymbol{\theta}}$$

$$= \sum_{(f,e)} -\delta_{(f,e)} \frac{\partial \text{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e)}{\partial \boldsymbol{\theta}} \quad (7)$$

which takes the form of summation over all phrase pairs occurring either in a training sample (stochastic mode) or in the entire training data (batch mode). $\delta_{(f,e)}$ in (7) is known as the *error* term of the phrase pair (f, e), and is defined as

$$\delta_{(f,e)} = -\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \text{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e)} \quad (8)$$

It describes how the overall loss changes with the translation score of the phrase pair (f, e). We will leave the derivation of $\delta_{(f,e)}$ to Section 5.1.2, and will first describe how the gradient of $\text{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e)$ w.r.t. $\boldsymbol{\theta}$ is computed.

5.1.1 Computing $\partial \text{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e)/\partial \boldsymbol{\theta}$

Without loss of generality, we use the following notations to describe a neural network:

- $\mathbf{W}_l$ is the projection matrix for the *l*-th layer of the neural network;
- $\mathbf{x}$ is the input word vector of a phrase;
- $\mathbf{z}^l$ is the sum vector of the *l*-th layer; and
- $\mathbf{y}^l = \sigma(\mathbf{z}^l)$ is the output vector of the *l*-th layer, where σ is an activation function;

Thus, the CPTM defined by (2) and (3) can be represented as

$$\mathbf{z}^1 = {\mathbf{W}_1}^{\mathrm{T}}\mathbf{x}$$
$$\mathbf{y}^1 = \sigma(\mathbf{z}^1)$$
$$\mathbf{z}^2 = {\mathbf{W}_2}^{\mathrm{T}}\mathbf{y}^1$$
$$\mathbf{y}^2 = \sigma(\mathbf{z}^2)$$
$$\text{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e) = \left(\mathbf{y}_f^2\right)^{\mathrm{T}}\mathbf{y}_e^2$$

The gradient of the matrix $\mathbf{W}_2$ which projects the hidden vector to the output vector is computed as:

$$\frac{\partial \text{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e)}{\partial \mathbf{W}_2} = \frac{\partial \left(\mathbf{y}_f^2\right)^{\mathrm{T}}}{\partial \mathbf{W}_2}\mathbf{y}_e^2 + \left(\mathbf{y}_f^2\right)^{\mathrm{T}} \frac{\partial \mathbf{y}_e^2}{\partial \mathbf{W}_2}$$

$$= \mathbf{y}_f^1 \left(\mathbf{y}_e^2 \circ \sigma'\left(\mathbf{z}_f^2\right)\right)^{\mathrm{T}} + \mathbf{y}_e^1 \left(\mathbf{y}_f^2 \circ \sigma'\left(\mathbf{z}_e^2\right)\right)^{\mathrm{T}} \quad (9)$$

where $\circ$ is the element-wise multiplication (Hadamard product). Applying the back propagation principle, the gradient of the projection matrix mapping the input vector to the hidden vector $\mathbf{W}_1$ is computed as

$$\frac{\partial \text{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e)}{\partial \mathbf{W}_1}$$
$$= \mathbf{x}_f \left(\mathbf{W}_2 \left(\mathbf{y}_e^2 \circ \sigma'\left(\mathbf{z}_f^2\right)\right) \circ \sigma'\left(\mathbf{z}_f^1\right)\right)^{\mathrm{T}}$$
$$+ \mathbf{x}_e \left(\mathbf{W}_2 \left(\mathbf{y}_f^2 \circ \sigma'\left(\mathbf{z}_e^2\right)\right) \circ \sigma'\left(\mathbf{z}_e^1\right)\right)^{\mathrm{T}} \quad (10)$$

The derivation can be easily extended to a neural network with multiple hidden layers.

5.1.2 Computing $\delta_{(f,e)}$

To simplify the notation, we rewrite our loss function of (5) and (6) over one training sample as

$$\mathcal{L}(\boldsymbol{\theta}) = -\mathrm{xBleu}(\boldsymbol{\theta}) = -\frac{\mathrm{G}(\boldsymbol{\theta})}{\mathrm{Z}(\boldsymbol{\theta})} \quad (11)$$

where

$$\mathrm{G}(\boldsymbol{\theta}) = \sum_E \mathrm{sBleu}(E, E_i) \exp\left(\boldsymbol{\lambda}^{\mathrm{T}} \mathbf{h}(F_i, E, A)\right)$$

$$\mathrm{Z}(\boldsymbol{\theta}) = \sum_E \exp\left(\boldsymbol{\lambda}^{\mathrm{T}} \mathbf{h}(F_i, E, A)\right)$$

Combining (8) and (11), we have

$$\delta_{(f,e)} = \frac{\partial \mathrm{xBleu}(\boldsymbol{\theta})}{\partial \mathrm{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e)} \quad (12)$$

$$= \frac{1}{\mathrm{Z}(\boldsymbol{\theta})} \left(\frac{\partial \mathrm{G}(\boldsymbol{\theta})}{\partial \mathrm{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e)} - \frac{\partial \mathrm{Z}(\boldsymbol{\theta})}{\partial \mathrm{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e)} \mathrm{xBleu}(\boldsymbol{\theta}) \right)$$

Because $\boldsymbol{\theta}$ is only relevant to h_{M+1} which is defined in (4), we have

$$\frac{\partial \boldsymbol{\lambda}^{\mathrm{T}} \mathbf{h}(F_i, E, A)}{\partial \mathrm{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e)} = \lambda_{M+1} \frac{\partial h_{M+1}(F_i, E, A)}{\partial \mathrm{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e)}$$

$$= \lambda_{M+1} N(f, e; A) \quad (13)$$

where $N(f, e; A)$ is the number of times the phrase pair (f, e) occurs in A. Combining (12) and (13), we end up with the following equation

$$\delta_{(f,e)} = \sum_{(E,A) \in GEN(F_i)} \mathrm{U}(\boldsymbol{\theta}, E) P(E|F_i) \lambda_{M+1} N(f, e; A) \quad (14)$$

where

$$\mathrm{U}(\boldsymbol{\theta}, E) = \mathrm{sBleu}(E_i, E) - \mathrm{xBleu}(\boldsymbol{\theta}).$$

5.2 The Training Algorithm

In our experiments we train the parameters of the CPTM, $\boldsymbol{\theta}$, using the L-BFGS optimizer described in Andrew and Gao (2007), together with the loss function described in (5). The gradient is computed as described in Sections 5.1. Although SGD has been advocated for neural network training due to its simplicity and its robustness to local minima (Bengio 2009), we find that in our task that the L-BFGS minimizes the loss in a desirable fashion empirically when iterating over the complete training data (batch mode). For example, the convergence of the algorithm was found to be smooth, despite the non-convexity in our loss. Another merit of batch training is that the gradient over all training data can be computed efficiently. As shown in Section 5.1, computing $\partial \mathrm{sim}_{\theta}(\mathrm{x}_f, \mathrm{x}_e)/\partial\theta$ requires large-scale matrix multiplications, and is expensive for multi-layer neural networks. Eq. (7) suggests that $\partial \mathrm{sim}_{\theta}(\mathrm{x}_f, \mathrm{x}_e)/\partial\theta$ and $\delta_{(f,e)}$ can be computed separately, thus making the computation cost of the former term only depends on the number of phrase pairs in the phrase table, but not the size of training data. Therefore, the training method described here can be used on larger amounts of training data with little difficulty.

As described in Section 4, we take three steps to learn the parameters for both the log-linear model of SMT and the CPTM. While steps 1 and 3 can be easily parallelized on a computer cluster, the CPTM training is performed on a single machine. For example, given a phrase table containing 16M pairs and a 1M-sentence training set, it takes a couple of hours to generate the N-best lists on a cluster, and about 10 hours to train the CPTM on a Xeon E5-2670 2.60GHz machine.

For a non-convex problem, model initialization is important. In our experiments we always initialize $\mathbf{W}_1$ using a bilingual topic model trained on parallel data (see detail in Section 6.2), and $\mathbf{W}_2$ as an identity matrix. In principle, the loss function of (5) can be further regularized (e.g. by adding a term of L_2 norm) to deal with overfitting. However, we did not find clear empirical advantage over the simpler early stop approach in a pilot study, which is adopted in the experiments in this paper.

6 Experiments

This section evaluates the CPTM presented on two translation tasks using WMT data sets. We first describe the data sets and baseline setup. Then we present experiments where we compare different versions of the CPTM and previous models.

6.1 Experimental Setup

Baseline. We experiment with an in-house phrase-based system similar to Moses (Koehn et al. 2007), where the translation candidates are scored by a set of common features including maximum likelihood estimates of source given target phrase mappings $P_{MLE}(e|f)$ and vice versa $P_{MLE}(f|e)$, as well as lexical weighting estimates $P_{LW}(e|f)$ and $P_{LW}(f|e)$, word and phrase penalties, a linear distortion feature, and a lexicalized reordering feature. The baseline includes a standard 5-gram modified Kneser-Ney language model trained on the target side of the parallel corpora described below. Log-linear weights are estimated with the MERT algorithm (Och 2003).

Evaluation. We test our models on two different data sets. First, we train an English to French system based on the data of WMT 2006 shared task (Koehn and Monz 2006). The parallel corpus includes 688K sentence pairs of parliamentary proceedings for training. The development set contains 2000 sentences, and the test set contains other 2000 sentences, all from the official WMT 2006 shared task.

Second, we experiment with a French to English system developed using 2.1M sentence pairs of training data, which amounts to 102M words, from the WMT 2012 campaign. The majority of the training data set is parliamentary proceedings except for 5M words which are newswire. We use the 2009 newswire data set, comprising 2525 sentences, as the development set. We evaluate on four newswire domain test sets from 2008, 2010 and 2011 as well as the 2010 system combination test set, containing 2034 to 3003 sentences.

In this study we perform a detailed empirical comparison using the WMT 2006 data set, and verify our best models and results using the larger WMT 2012 data set.

The metric used for evaluation is case insensitive BLEU score (Papineni et al. 2002). We also perform a significance test using the Wilcoxon signed rank test. Differences are considered statistically significant when the *p*-value is less than 0.05.

6.2 Results of the CPTM

Table 1 shows the results measured in BLEU evaluated on the WMT 2006 data set, where Row 1 is the baseline system. Rows 2 to 4 are the systems enhanced by integrating different versions of the CPTM. Rows 5 to 7 present the results of previous models. Row 8 is our best system. Table 2 shows the main results on the WMT 2012 data set.

CPTM is the model described in Sections 4. As illustrated in Figure 2, the number of the nodes in the input layer is the vocabulary size d. Both the hidden layer and the output layer have 100 nodes[6]. That is, $\mathbf{W}^1$ is a $d \times 100$ matrix and $\mathbf{W}^2$ a 100×100 matrix. The result shows that **CPTM** leads to a substantial improvement over the baseline system with a statistically significant margin of 1.0 BLEU points as in Table 1.

We have developed a set of variants of **CPTM** to investigate two design choices we made in developing the CPTM: (1) whether to use a linear projection or a multi-layer nonlinear projection; and (2) whether to compute the phrase similarity using word-word similarities as suggested by e.g., the lexical weighting model (Koehn et al. 2003). We compare these variants on the WMT 2006 data set, as shown in Table 1.

#	Systems	WMT test2006
1	**Baseline**	33.06
2	**CPTM**	**34.10**$^{\alpha}$
3	**CPTM$_{\mathbf{L}}$**	33.60$^{\alpha\beta}$
4	**CPTM$_{\mathbf{W}}$**	33.25$^{\beta}$
5	**BLTM$_{\mathbf{PR}}$**	33.15$^{\beta}$
6	**DPM**	33.29$^{\beta}$
7	**MRF$_{\mathbf{P}}$**	33.91$^{\alpha}$
8	**Comb (2 + 7)**	**34.39**$^{\alpha\beta}$

Table 1: BLEU results for the English to French task using translation models and systems built on the WMT 2006 data set. The superscripts α and β indicate statistically significant difference ($p < 0.05$) from **Baseline** and **CPTM**, respectively.

CPTM$_{\mathbf{L}}$ (Row 3 in Table 1) uses a linear neural network to project a word vector of a phrase $\mathbf{x}$ to a feature vector $\mathbf{y}$: $\mathbf{y} \equiv \phi(\mathbf{x}) = \mathbf{W}^{\mathrm{T}}\mathbf{x}$, where $\mathbf{W}$ is a $d \times 100$ projection matrix. The translation score of a source phrase f and a target phrase e is measured as the similarity of their feature vectors. We choose cosine similarity because it works better than dot product for linear projection.

CPTM$_{\mathbf{W}}$ (Row 4 in Table 1) computes the phrase similarity using word-word similarity scores. This follows the common smoothing strategy of addressing the data sparseness problem in modeling phrase translations, such as the lexical weighting model (Koehn et al. 2003) and the word factored n-gram translation model (Son et al. 2012). Let w denote a word, and f and e the source and target phrases, respectively. We define

$$\mathrm{sim}(f,e) = \frac{1}{|f|}\sum_{w\in f}\mathrm{sim}_\tau(w,e) + \frac{1}{|e|}\sum_{w\in e}\mathrm{sim}_\tau(w,f)$$

where $\mathrm{sim}_\tau(w,e)$ (or $\mathrm{sim}_\tau(w,f)$) is the word-phrase similarity, and is defined as a smooth approximation of the maximum function

$$\mathrm{sim}_\tau(w,e) = \frac{\sum_{w'\in e}\mathrm{sim}(w,w')\exp\big(\tau\mathrm{sim}(w,w')\big)}{\sum_{w'\in e}\exp\big(\tau\mathrm{sim}(w,w')\big)}$$

[6] We can achieve slightly better results using more nodes in the hidden and output layers, say 500 nodes. But the model training is too slow to perform a detailed study within a reasonable time. Therefore, all the models reported in this paper use 100 nodes.

#	Systems	dev	news2011	news2010	news2008	newssyscomb2010
1	**Baseline**	23.58	25.24	24.35	20.36	24.14
2	**MRF$_P$**	24.07$^{\alpha}$	26.00$^{\alpha}$	24.90	20.84$^{\alpha}$	25.05$^{\alpha}$
3	**CPTM**	24.12$^{\alpha}$	26.25$^{\alpha}$	25.05$^{\alpha}$	21.15$^{\alpha\beta}$	24.91$^{\alpha}$
4	**Comb (2 + 3)**	**24.46**$^{\alpha\beta}$	**26.56**$^{\alpha\beta}$	**25.52**$^{\alpha\beta}$	**21.64**$^{\alpha\beta}$	**25.22**$^{\alpha}$

Table 2: BLEU results for the French to English task using translation models and systems built on the WMT 2012 data set. The superscripts α and β indicate statistically significant difference ($p < 0.05$) from **Baseline** and **MRF$_p$**, respectively.

where $\mathrm{sim}_\tau(w, e)$ (or $\mathrm{sim}_\tau(w, f)$) is the word-phrase similarity, and is defined as a smooth approximation of the maximum function
where τ is the tuned smoothing parameter.

Similar to **CPTM**, **CPTM$_W$** also uses a nonlinear projection to map each word (not a phrase vector as in **CPTM**) to a feature vector.

Two observations can be made by comparing **CPTM** in Row 2 to its variants in Table 1. First of all, it is more effective to model the phrase translation directly than decomposing it into word-word translations in the CPTMs. Second, we see that the nonlinear projection is able to generate more effective features, leading to better results than the linear projection.

We also compare the best version of the CPTM i.e., **CPTM**, with three related models proposed previously. We start the discussion with the results on the WMT 2006 data set in Table 1.

Rows 5 and 6 in Table 1 are two state-of-the-art latent semantic models that are originally trained on clicked query-document pairs (i.e., clickthrough data extracted from search logs) for query-document matching (Gao et al. 2011). To adopt these models for SMT, we view source-target sentence pairs as clicked query-document pairs, and trained both models using the same methods as in Gao et al. (2011) on the parallel bilingual training data described earlier. Specifically, **BTLM$_{PR}$** is an extension to PLSA, and is the best performer among different versions of the Bi-Lingual Topic Model (BLTM) described in Gao et al. (2011). BLTM with Posterior Regularization (**BLTM$_{PR}$**) is trained on parallel training data using the EM algorithm with a constraint enforcing a source sentence and its paralleled target sentence to not only share the same prior topic distribution, but to also have similar fractions of words assigned to each topic. We incorporated the model into the log-linear model for SMT (1) as follows. First of all, the topic distribution of a source sentence F_i, denoted by $P(z|F_i)$, is induced from the learned topic-word distributions using EM. Then, each translation candidate E in the N-best list $\mathrm{GEN}(F_i)$ is scored as

$$P(E|F_i) = \prod_{w \in E} \sum_z P(w|z) P(z|F_i)$$

$P(F_i|E)$ can be similarly computed. Finally, the logarithms of the two probabilities are incorporated into the log-linear model of (1) as two additional features. **DPM** is the Discriminative Projection Model described in Gao et al. (2011), which is an extension of LSA. **DPM** uses a matrix to project a word vector of a sentence to a feature vector. The projection matrix is learned on parallel training data using the S2Net algorithm (Yih et al. 2011). **DPM** can be incorporated into the log-linear model for SMT (1) by introducing a new feature h_{M+1} for each phrase pair, which is defined as the cosine similarity of the phrases in the project space.

As we see from Table 1, both latent semantic models, although leading to some slight improvement over **Baseline**, are much less effective than **CPTM**.

Finally, we compare the CPTM with the Markov Random Field model using phrase features (**MRF$_P$** in Tables 1 and 2), proposed by Gao and He (2013)[7], on both the WMT 2006 and WMT 2012 datasets. **MRF$_p$** is a state-of-the-art large scale discriminative training model that uses the same expected BLEU training criterion, which has proven to give superior performance across a range of MT tasks recently (He and Deng 2012, Setiawan and Zhou 2013, Gao and He 2013).

Unlike **CPTM**, **MRF$_p$** is a linear model that simply treats each phrase pair as a single feature. Therefore, although both are trained using the

[7] Gao and He (2013) reported results of MRF models with different feature sets. We picked the MRF using phrase features only (**MRF$_P$**) for comparison since we are mainly interested in phrase representation.

same expected BLEU based objective function, **CPTM** and **MRF$_p$** model the translation relationship between two phrases from different angles. **MRF$_p$** estimates one translation score for each phrase pair explicitly without parameter sharing, while in **CPTM**, all phrases share the same neural network that projects raw phrases to the continuous space, providing a more smoothed estimation of the translation score for each phrase pair.

The results in Tables 1 and 2 show that **CPTM** outperforms **MRF$_P$** on most of the test sets across the two WMT data sets, but the difference between them is often not significant. Our interpretation is that although **CPTM** provides a better smoothed estimation for low-frequent phrase pairs, which otherwise suffer the data sparsity issue, **MRF$_p$** provides a more precise estimation for those high-frequent phrase pairs. That is, **CPTM** and **MRF$_p$** capture complementary information for translation. We thus combine **CPTM** and **MRF$_P$** (**Comb** in Tables 1 and 2) by incorporating two features, each for one model, into the log-linear model of SMT (1). We observe that for both translation tasks, accuracy improves by up to 0.8 BLEU over **MRF$_P$** alone (e.g., on the news2008 test set in Table 2). The results confirm that **CPTM** captures complementary translation information to **MRF$_p$**. Overall, we improve accuracy by up to 1.3 BLEU over the baseline on both WMT data sets.

7 Conclusions

The work presented in this paper makes two major contributions. First, we develop a novel phrase translation model for SMT, where joint representations are exploited of a phrase in the source language and of its translation in the target language, and where the translation score of the pair of source-target phrases are represented as the distance between their feature vectors in a low-dimensional, continuous space. The space is derived from the representations generated using a multilayer neural network. Second, we present a new learning method to train the weights in the multilayer neural network for the end-to-end BLEU metric directly. The training method is based on L-BFGS. We describe in detail how the gradient in closed form, as required for efficient optimization, is derived. The objective function, which takes the form of the expected BLEU computed from N-best lists, is very different from the usual objective functions used in most existing architectures of neural networks, e.g., cross entropy (Hinton et al. 2012) or mean square error (Deng et al. 2012). We hence have provided details in the derivation of the gradient, which can serve as an example to guide the derivation of neural network learning with other non-standard objective functions in the future.

Our evaluation on two WMT data sets show that incorporating the continuous-space phrase translation model into the log-linear framework significantly improves the accuracy of a state-of-the-art phrase-based SMT system, leading to a gain up to 1.3 BLEU. Careful implementation of the L-BFGS optimization based on the BLEU-centric objective function, together with the associated closed-form gradient, is a key to the success.

A natural extension of this work is to expand the model and learning algorithm from shallow to deep neural networks. The deep models are expected to produce more powerful and flexible semantic representations (e.g., Tur et al., 2012), and thus greater performance gain than what is presented in this paper.

8 Acknowledgements

We thank Michael Auli for providing a dataset and for helpful discussions. We also thank the four anonymous reviewers for their comments.

References

Andrew, G. and Gao, J. 2007. Scalable training of L1-regularized log-linear models. In *ICML*.

Auli, M., Galley, M., Quirk, C. and Zweig, G. 2013 Joint language and translation modeling with recurrent neural networks. In *EMNLP*.

Bengio, Y. 2009. Learning deep architectures for AI. *Fundamental Trends Machine Learning*, vol. 2, no. 1, pp. 1–127.

Bengio, Y., Duharme, R., Vincent, P., and Janvin, C. 2003. A neural probabilistic language model. *JMLR*, 3:1137-1155.

Blei, D. M., Ng, A. Y., and Jordan, M. J. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3: 993-1022.

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

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

DeNero, J., Gillick, D., Zhang, J., and Klein, D. 2006. Why generative phrase models underperform surface heuristics. In *Workshop on Statistical Machine Translation*, pp. 31-38.

Deng, L., Yu, D., and Platt, J. 2012. Scalable stacking and learning for building deep architectures. In *ICASSP*.

Diamantaras, K. I., and Kung, S. Y. 1996. *Principle Component Neural Networks: Theory and Applications.* Wiley-Interscience.

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

Ganchev, K., Graca, J., Gillenwater, J., and Taskar, B. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11 (2010): 2001-2049.

Gao, J., and He, X. 2013. Training MRF-based translation models using gradient ascent. In *NAACL-HLT*, pp. 450-459.

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

He, X., and Deng, L. 2012. Maximum expected bleu training of phrase and lexicon translation models. In *ACL*, pp. 292-301.

Hinton, G., and Salakhutdinov, R., 2010. Discovering Binary Codes for Documents by Learning Deep Generative Models. *Topics in Cognitive Science*, pp. 1-18.

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., and Kingsbury, B., 2012. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82-97.

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

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

Kalchbrenner, N. and Blunsom, P. 2013. Recurrent continuous translation models. In *EMNLP*.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. 2007. Moses: open source toolkit for statistical machine translation. In *ACL 2007*, demonstration session.

Koehn, P. and Monz, C. 2006. Manual and automatic evaluation of machine translation between European languages. In *Workshop on Statistical Machine Translation*, pp. 102-121.

Koehn, P., Och, F., and Marcu, D. 2003. Statistical phrase-based translation. In *HLT-NAACL*, pp. 127-133.

Lambert, P. and Banchs, R. E. 2005. Data inferred multi-word expressions for statistical machine translation. In *MT Summit X*, Phuket, Thailand.

Li, P., Liu, Y., and Sun, M. 2013. Recursive autoencoders for ITG-based translation. In *EMNLP*.

Liang,P., Bouchard-Cote,A., Klein, D. and Taskar, B. 2006. An end-to-end discriminative approach to machine translation. In *COLING-ACL*.

Marcu, D., and Wong, W. 2002. A phrase-based, joint probability model for statistical machine translation. In *EMNLP*.

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

Mikolov, T., Kombrink, S., Burget, L., Cernocky, J., and Khudanpur, S. 2011. Extensions of recurrent neural network language model. In *ICASSP*, pp. 5528-5531.

Mikolov, T. 2012. Statistical Language Model based on Neural Networks. *Ph.D. thesis*, Brno University of Technology.

Mikolov, T., Le, Q. V., and Sutskever, H. 2013a. Exploiting similarities among languages for machine translation. CoRR. 2013; abs/1309.4148.

Mikolov, T., Yih, W. and Zweig, G. 2013b. Linguistic Regularities in Continuous Space Word Representations. In *NAACL-HLT*.

Mimno, D., Wallach, H., Naradowsky, J., Smith, D. and McCallum, A. 2009. Polylingual topic models. In *EMNLP*.

Niehues J., Herrmann, T., Vogel, S., and Waibel, A. 2011. Wider context by using bilingual language models in machine translation.

Och, F. 2003. Minimum error rate training in statistical machine translation. In *ACL*, pp. 160-167.

Och, F., and Ney, H. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 29(1): 19-51.

Papineni, K., Roukos, S., Ward, T., and Zhu W-J. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL*.

Platt, J., Toutanova, K., and Yih, W. 2010. Translingual Document Representations from Discriminative Projections. In *EMNLP*.

Rosti, A-V., Hang, B., Matsoukas, S., and Schwartz, R. S. 2011. Expected BLEU training for graphs: bbn system description for WMT system combination task. In *Workshop on Statistical Machine Translation*.

Schwenk, H., Costa-Jussa, M. R. and Fonollosa, J. A. R. 2007. Smooth bilingual n-gram translation. In *EMNLP-CoNLL*, pp. 430-438.

Schwenk, H. 2012. Continuous space translation models for phrase-based statistical machine translation. In *COLING*.

Schwenk, H., Rousseau, A., and Mohammed A. 2012. Large, pruned or continuous space language models on a GPU for statistical machine translation. In *NAACL-HLT Workshop on the future of language modeling for HLT*, pp. 11-19.

Setiawan, H. and Zhou, B., 2013. Discriminative training of 150 million translation parameters and its application to pruning. In *NAACL*.

Socher, R., Huval, B., Manning, C., Ng, A., 2012. Semantic Compositionality through Recursive Matrix-Vector Spaces. In *EMNLP*.

Socher, R., Lin, C., Ng, A. Y., and Manning, C. D. 2011. Parsing natural scenes and natural language with recursive neural networks. In *ICML*.

Son, L. H., Allauzen, A., and Yvon, F. 2012. Continuous space translation models with neural networks. In *NAACL-HLT*, pp. 29-48.

Sundermeyer, M., Oparin, I., Gauvain, J-L. Freiberg, B., Schluter, R. and Ney, H. 2013. Comparison of feed forward and recurrent neural network language models. In *ICASSP*, pp. 8430–8434.

Tur, G, Deng, L., Hakkani-Tur, D., and He, X., 2012. Towards deeper understanding: deep convex networks for semantic utterance classification. In *ICASSP*.

Vinokourov,A., Shawe-Taylor,J. and Cristianini,N. 2002. Inferring a semantic representation of text via cross-language correlation analysis. In *NIPS*.

Weston, J., Bengio, S., and Usunier, N. 2011. Large scale image annotation: learning to rank with joint word-image embeddings. In *IJCAI*.

Wuebker, J., Mauser, A., and Ney, H. 2010. Training phrase translation models with leaving-one-out. In *ACL*, pp. 475-484.

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

Zhang, Y., Deng, L., He, X., and Acero, A. 2011. A novel decision function and the associated decision-feedback learning for speech translation. In *ICASSP*.

Zhila, A., Yih, W., Meek, C., Zweig, G. and Mikolov, T. 2013. Combining heterogeneous models for measuring relational similarity. In *NAACL-HLT*.

Zou, W. Y., Socher, R., Cer, D., and Manning, C. D. 2013. Bilingual word embeddings for phrase-based machine translation. In *EMNLP*.

Adaptive Quality Estimation for Machine Translation

Marco Turchi[1] **Antonios Anastasopoulos**[3]
José G. C. de Souza[1,2] **Matteo Negri**[1]
[1] FBK - Fondazione Bruno Kessler, Via Sommarive 18, 38123 Trento, Italy
[2] University of Trento, Italy
[3] National Technical University of Athens, Greece
{turchi,desouza,negri}@fbk.eu
anastasopoulos.ant@gmail.com

Abstract

The automatic estimation of machine translation (MT) output quality is a hard task in which the selection of the appropriate algorithm and the most predictive features over reasonably sized training sets plays a crucial role. When moving from controlled lab evaluations to real-life scenarios the task becomes even harder. For current MT quality estimation (QE) systems, additional complexity comes from the difficulty to model user and domain changes. Indeed, the instability of the systems with respect to data coming from different distributions calls for adaptive solutions that react to new operating conditions. To tackle this issue we propose an online framework for adaptive QE that targets reactivity and robustness to user and domain changes. Contrastive experiments in different testing conditions involving user and domain changes demonstrate the effectiveness of our approach.

1 Introduction

After two decades of steady progress, research in statistical machine translation (SMT) started to cross its path with translation industry with tangible mutual benefit. On one side, SMT research brings to the industry improved output quality and a number of appealing solutions useful to increase translators' productivity. On the other side, the market needs suggest concrete problems to solve, providing real-life scenarios to develop and evaluate new ideas with rapid turnaround. The evolution of computer-assisted translation (CAT) environments is an evidence of this trend, shown by the increasing interest towards the integration of suggestions obtained from MT engines with those derived from translation memories (TMs).

The possibility to speed up the translation process and reduce its costs by post-editing good-quality MT output raises interesting research challenges. Among others, these include deciding *what* to present as a suggestion, and *how* to do it in the most effective way.

In recent years, these issues motivated research on automatic QE, which addresses the problem of estimating the quality of a translated sentence given the source and without access to reference translations (Blatz et al., 2003; Specia et al., 2009; Mehdad et al., 2012). Despite the substantial progress done so far in the field and in successful evaluation campaigns (Callison-Burch et al., 2012; Bojar et al., 2013), focusing on concrete market needs makes possible to further define the scope of research on QE. For instance, moving from controlled lab testing scenarios to real working environments poses additional constraints in terms of adaptability of the QE models to the variable conditions of a translation job. Such variability is due to two main reasons:

1. **The notion of MT output quality is highly subjective** (Koponen, 2012; Turchi et al., 2013; Turchi and Negri, 2014). Since the quality standards of individual users may vary considerably (*e.g.* according to their knowledge of the source and target languages), the estimates of a static QE model trained with data collected from a group of post-editors might not fit with the actual judgements of a new user;

2. **Each translation job has its own specificities** (domain, complexity of the source text, average target quality). Since data from a new job may differ from those used to train the QE model, its estimates on the new instances might result to be biased or uninformative.

The ability of a system to self-adapt to the be-

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 710–720,
Baltimore, Maryland, USA, June 23-25 2014.

haviour of specific users and domain changes is a facet of the QE problem that so far has been disregarded. To cope with these issues and deal with the erratic conditions of real-world translation workflows, we propose **an adaptive approach to QE** that is sensitive and robust to differences between training and test data. Along this direction, our main contribution is a framework in which QE models can be trained and can continuously evolve over time accounting for knowledge acquired from post editors' work.

Our approach is **based on the online learning paradigm** and exploits a key difference between such framework and the batch learning methods currently used. On one side, the QE models obtained with batch methods are learned exclusively from a predefined set of training examples under the assumption that they have similar characteristics with respect to the test data. This makes them suitable for controlled evaluation scenarios where such condition holds. On the other side, online learning techniques are designed to learn in a stepwise manner (either from scratch, or by refining an existing model) from new, unseen test instances by taking advantage of external feedback. This makes them suitable for real-life scenarios where the new instances to be labelled can considerably differ from the data used to train the QE model.

To develop our approach, different online algorithms have been embedded in the backbone of a QE system. This required the adaptation of its standard batch learning workflow to:

1. Perform online feature extraction from a source–target pair (*i.e.* one instance at a time instead of processing an entire training set);
2. Emit a prediction for the input instance;
3. Gather user feedback for the instance (*i.e.* calculating a "true label" based on the amount of user post-editions);
4. Send the true label back to the model to update its predictions for future instances.

Focusing on the adaptability to user and domain changes, we report the results of comparative experiments with two online algorithms and the standard batch approach. The evaluation is carried out by measuring the global error of each algorithm on test sets featuring different degrees of similarity with the data used for training. Our results show that the sensitivity of online QE models to different distributions of training and test instances makes them more suitable than batch methods for integration in a CAT framework.

Our adaptive QE infrastructure has been released as open source. Its C++ implementation is available at `http://hlt.fbk.eu/technologies/aqet`.

2 Related work

QE is generally cast as a supervised machine learning task, where a model trained from a collection of (*source, target, label*) instances is used to predict labels[1] for new, unseen test items (Specia et al., 2010).

In the last couple of years, research in the field received a strong boost by the shared tasks organized within the WMT workshop on SMT,[2] which is also the framework of our first experiment in §5. Current approaches to the tasks proposed at WMT have mainly focused on three main directions, namely: *i)* feature engineering, as in (Hardmeier et al., 2012; de Souza et al., 2013a; de Souza et al., 2013b; Rubino et al., 2013b), *ii)* model learning with a variety of classification and regression algorithms, as in (Bicici, 2013; Beck et al., 2013; Soricut et al., 2012), and *iii)* feature selection as a way to overcome sparsity and overfitting issues, as in (Soricut et al., 2012).

Being optimized to perform well on specific WMT sub-tasks and datasets, current systems reflect variations along these directions but leave important aspects of the QE problem still partially investigated or totally unexplored.[3] Among these, the necessity to model the diversity of human quality judgements and correction strategies (Koponen, 2012; Koponen et al., 2012) calls for solutions that: *i)* account for annotator-specific behaviour, thus being capable of learning from inherently noisy datasets produced by multiple annotators, and *ii)* self-adapt to changes in data distribution, learning from user feedback on new, unseen test items.

[1] Possible label types include *post-editing effort scores* (*e.g.* 1-5 Likert scores indicating the estimated percentage of MT output that has to be corrected), *HTER values* (Snover et al., 2006), and post-editing *time* (*e.g.* seconds per word).

[2] `http://www.statmt.org/wmt13/`

[3] For a comprehensive overview of the QE approaches proposed so far we refer the reader to the WMT12 and WMT13 QE shared task reports (Callison-Burch et al., 2012; Bojar et al., 2013).

These interconnected issues are particularly relevant in the CAT framework, where translation jobs from different domains are routed to professional translators with different idiolect, background and quality standards.

The first aspect, modelling annotators' individual behaviour and interdependences, has been addressed by Cohn and Specia (2013), who explored multi-task Gaussian Processes as a way to jointly learn from the output of multiple annotations. This technique is suitable to cope with the unbalanced distribution of training instances and yields better models when heterogeneous training datasets are available.

The second problem, the adaptability of QE models, has not been explored yet. A common trait of all current approaches, in fact, is the reliance on batch learning techniques, which assume a "static" nature of the world where new unseen instances that will be encountered will be similar to the training data.[4] However, similarly to translation memories that incrementally store translated segments and evolve over time incorporating users style and terminology, all components of a CAT tool (the MT engine and the mechanisms to assign quality scores to the suggested translations) should take advantage of translators feedback.

On the MT system side, research on adaptive approaches tailored to interactive SMT and CAT scenarios explored the online learning protocol (Littlestone, 1988) to improve various aspects of the decoding process (Cesa-Bianchi et al., 2008; Ortiz-Martínez et al., 2010; Martínez-Gómez et al., 2011; Martínez-Gómez et al., 2012; Mathur et al., 2013; Bertoldi et al., 2013).

As regards QE models, our work represents the first investigation on incremental adaptation by exploiting users feedback to provide targeted (system, user, or project specific) quality judgements.

3 Online QE for CAT environments

When operating with advanced CAT tools, translators are presented with suggestions (either matching fragments from a translation memory or automatic translations produced by an MT system) for each sentence of a source document. Before being approved and published, translation suggestions may require different amounts of post-editing operations depending on their quality.

[4]This assumption holds in the WMT evaluation scenario, but it is not necessarily valid in real operating conditions.

Each post-edition brings a wealth of dynamic knowledge about the whole translation process and the involved actors. For instance, adaptive QE components could exploit information about the distance between automatically assigned scores and the quality standards of individual translators (inferred from the amount of their corrections) to "profile" their behaviour.

The online learning paradigm fits well with this research objective. In the online framework, differently from the batch mode, the learning algorithm sequentially processes an unknown sequence of instances $X = x_1, x_2, ..., x_n$, returning a prediction $p(x_i)$ as output at each step. Differences between $p(x_i)$ and the true label $\hat{p}(x_i)$ obtained as feedback are used by the learner to refine the next prediction $p(x_{i+1})$.

In our experiments on adaptive QE we aim to predict the quality of the suggested translations in terms of HTER, which measures the minimum edit distance between the MT output and its manually post-edited version in the [0,1] interval.[5] In this scenario:

- The set of instances X is represented by (*source, target*) pairs;
- The prediction $p(x_i)$ is the automatically estimated HTER score;
- The true label $\hat{p}(x_i)$ is the actual HTER score calculated over the target and its post-edition.

At each step of the process, the goal of the learner is to exploit user post-editions to reduce the difference between the predicted HTER values and the true labels for the following (*source, target*) pairs.

As depicted in Figure 1, this is done as follows:

1. At step *i*, an unlabelled (*source, target*) pair x_i is sent to a feature extraction component. To this aim, we used an adapted version (Shah et al., 2014) of the open-source QuEst[6] tool (Specia et al., 2013). The tool, which implements a large number of features proposed by participants in the WMT QE shared tasks, has been modified to process one sentence at a time as requested for integration in a CAT environment;

[5]Edit distance is calculated as the number of edits (word insertions, deletions, substitutions, and shifts) divided by the number of words in the reference. Lower HTER values indicate better translations.

[6]`http://www.quest.dcs.shef.ac.uk/`

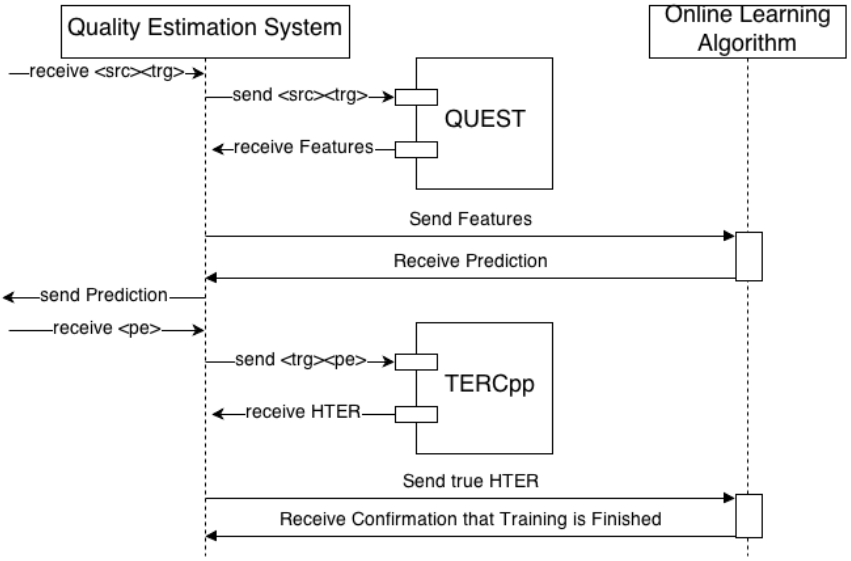


Figure 1: Online QE workflow. <src>, <trg> and <pe> respectively stand for the source sentence, the target translation and the post-edited target.

2. The extracted features are sent to an online regressor, which returns a QE prediction score $p(x_i)$ in the [0,1] interval (set to 0 at the first round of the iteration);

3. Based on the post-edition done by the user, the true HTER label $\hat{p}(x_i)$ is calculated by means of the TERCpp[7] open source tool;

4. The true label is sent back to the online algorithm for a stepwise model improvement. The updated model is then ready to process the following instance x_{i+1}.

This new paradigm for QE makes it possible to: *i)* let the QE system learn from one point at a time without complete re-training from scratch, *ii)* customize the predictions of an existing QE model with respect to a specific situation (post-editor or domain), or even *iii)* build a QE model from scratch when training data is not available.

For the sake of clarity it is worth observing that, at least in principle, a model built in a batch fashion could also be adapted to new test data. For instance, this could be done by running periodic re-training routines once a certain amount of new labelled instances has been collected (*de facto* mimicking an online process). Such periodic updates, however, would not represent a viable solution in the CAT framework where post-editors' work cannot be slowed by time-consuming procedures to re-train core system components from scratch.

[7] goo.gl/nkh2rE

4 Evaluation framework

To measure the adaptation capability of different QE models, we experiment with a range of conditions defined by variable degrees of similarity between training and test data.

The degree of similarity depends on several factors: the MT engine used, the domain of the documents to be translated, and the post-editing style of individual translators. In our experiments, the degree of similarity is measured in terms of ΔHTER, which is computed as the absolute value of the difference between the average HTER of the training and test sets. Large values indicate a low similarity between training and test data and a more challenging scenario for the learning algorithms.

4.1 Experimental setup

In the range of possible evaluation scenarios, our experiments cover:

- One artificial setting (§5) obtained from the WMT12 QE shared task data, in which training/test instances are arranged to reflect homogeneous distributions of the HTER labels.

- Two settings obtained from data collected with a CAT tool in real working conditions, in which different facets of the adaptive QE problem interact with each other. In the first (`user_change`, §6.1), training and test data from the same domain are obtained from different users. In the sec-

ond (user+domain_change, §6.2), training and test data are obtained from different users and domains.

For each setting, we compare an *adaptive* and an *empty* model against a system trained in *batch* mode. The *adaptive* model is built on top of an existing model created from the training data and exploits the new test instances to refine its predictions in a stepwise manner. The *empty* model only learns from the test set, simulating the worst condition where training data is not available. The *batch* model is built by learning only from the training data and is evaluated on the test set without exploiting information from the test instances.

Each model is also compared against a common baseline for regression tasks, which is particularly relevant in settings featuring different data distributions between training and test sets. This baseline (μ henceforth) is calculated by labelling each instance of the test set with the mean HTER score of the training set. Previous works (Rubino et al., 2013a) demonstrated that its results can be particularly hard to beat.

4.2 Performance indicator and feature set

To measure the adaptability of our model to a given test set we compute the Mean Absolute Error (MAE), a metric for regression problems also used in the WMT QE shared tasks. The MAE is the average of the absolute errors $e_i = |f_i - y_i|$, where f_i is the prediction of the model and y_i is the true value for the i^{th} instance.

As our focus is on the algorithmic aspect, in all experiments we use the same feature set, which consists of the seventeen features proposed in (Specia et al., 2009). This feature set, fully described in (Callison-Burch et al., 2012), takes into account the complexity of the source sentence (*e.g.* number of tokens, number of translations per source word) and the fluency of the target translation (*e.g.* language model probabilities). The results of previous WMT QE shared tasks have shown that these baseline features are particularly competitive in the regression task (with only few systems able to beat them at WMT12).

4.3 Online algorithms

In our experiments we evaluate two online algorithms, OnlineSVR (Parrella, 2007)[8] and Passive-Aggressive Perceptron (Crammer et al., 2006),[9] by comparing their performance with a batch learning strategy based on the Scikit-learn implementation of Support Vector Regression (SVR).[10]

The choice of the OnlineSVR and Passive-Aggressive (OSVR and PA henceforth) is motivated by different considerations. From a **performance** point of view, as an adaptation of ϵ-SVR which proved to be one of the top performing algorithms in the regression QE tasks at WMT, OSVR seems to be the best candidate. For this reason, we use the online adaptation of ϵ-SVR proposed by (Ma et al., 2003). The goal of OnlineSVR is to find a way to add each new sample to one of three sets (support, empty, error) maintaining the consistency of a set of conditions known as Karush-Kuhn Tucker (KKT) conditions. For each new point, OSVR starts a cycle where the samples are moved across the three sets until the KKT conditions are verified and the new point is assigned to one of the sets. If the point is identified as a support vector, the parameters of the model are updated. This allows OSVR to benefit from the prediction capability of ϵ-SVR in an online setting.

From a **practical** point of view, providing the best trade off between accuracy and computational time (He and Wang, 2012), PA represents a good solution to meet the demand of efficiency posed by the CAT framework. For each instance *i*, after emitting a prediction and receiving the true label, PA computes the ϵ-insensitive hinge loss function. If its value is larger than the tolerance parameter (ϵ), the weights of the model are updated as much as the aggressiveness parameter C allows. In contrast with OSVR, which keeps track of the most important points seen in the past (support vectors), the update of the weights is done without considering the previously processed *i-1* instances. Although it makes PA faster than OSVR, this is a riskier strategy because it may lead the algorithm to change the model to adapt to outlier points.

5 Experiments with WMT12 data

The motivations for experiments with training and test data featuring homogeneous label distributions are twofold. First, since in this artificial scenario adaptation capabilities are not required for the QE component, batch methods operate in the ideal conditions (as training and test are indepen-

[8] http://www2.imperial.ac.uk/~gmontana/onlinesvr.htm

[9] https://code.google.com/p/sofia-ml/

[10] http://scikit-learn.org/

WMT Dataset								
Train	Test	Δ HTER	μ MAE	Batch MAE	Adaptive MAE	Adaptive Alg.	Empty MAE	Empty Alg.
200	754	0.39	13.7	**13.2**	**13.2***	OSVR	13.5*	OSVR
600	754	1.32	13.8	**12.7**	12.9*	OSVR	13.5*	OSVR
1500	754	1.22	13.8	**12.7**	12.8*	OSVR	13.5*	OSVR

Table 1: MAE of the best performing *batch*, *adaptive* and *empty* models on WMT12 data. Training sets of different size and the test set have been arranged to reflect homogeneous label distributions.

dent and identically distributed). This makes possible to obtain from batch models the best possible performance to compare with. Second, this scenario provides the fairest conditions for such comparison because, in principle, online algorithms are not favoured by the possibility to learn from the diversity of the test instances.

For our controlled experiments we use the WMT12 English-Spanish corpus, which consists of 2,254 source-target pairs (1,832 for training, 422 for test). The HTER labels for our regression task are calculated from the post-edited version and the target sentences provided in the dataset.

To avoid biases in the label distribution, the WMT12 training and test data have been merged, shuffled, and eventually separated to generate three training sets of different size (200, 600, and 1500 instances), and one test set with 754 instances. For each algorithm, the training sets are used for learning the QE models, optimizing parameters (*i.e.* C, ϵ, the kernel and its parameters for SVR and OSVR; tolerance and aggressiveness for PA) through grid search in 10-fold cross-validation.

Evaluation is carried out by measuring the performance of the *batch* (learning only from the training set), the *adaptive* (learning from the training set and adapting to the test set), and the *empty* (learning from scratch from the test set) models in terms of global MAE scores on the test set.

Table 1 reports the results achieved by the best performing algorithm for each type of model (*batch*, *adaptive*, *empty*). As can be seen, close MAE values show a similar behaviour for the three types of models.[11] With the same amount of training data, the performance of the batch and the adaptive models (in this case always obtained with OSVR) is almost identical. This demonstrates that, as expected, the online algorithms do not take advantage of test data with a label distribution similar to the training set. All the models outperform the baseline, even if the minimal differences confirm the competitiveness of such a simple approach.

Overall, these results bring some interesting indications about the behaviour of the different online algorithms. First, the good results achieved by the empty models (less than one MAE point separates them from the best ones built on the largest training set) suggest their high potential when training data are not available. Second, our results show that OSVR is always the best performing algorithm for the adaptive and empty models. This suggests a lower capability of PA to learn from instances similar to the training data.

6 Experiments with CAT data

To experiment with adaptive QE in more realistic conditions we used a CAT tool[12] to collect two datasets of (*source*, *target*, *post_edited target*) English-Italian tuples.The source sentences in the datasets come from two documents from different domains, respectively legal (L) and information technology (IT). The L document, which was extracted from a European Parliament resolution published on the EUR-Lex platform,[13] contains 164 sentences. The IT document, which was taken from a software user manual, contains 280 sentences. The source sentences were translated with two SMT systems built by training the Moses toolkit (Koehn et al., 2007) on parallel data from the two domains (about 2M sentences for IT and 1.5M for L). Post-editions were collected from eight professional translators (four for each document) operating with the CAT tool in real working conditions.

According to the way they are created, the two datasets allow us to evaluate the adaptability of different QE models with respect to user changes

[11] Results marked with the "*" symbol are NOT statistically significant compared to the corresponding batch model. The others are always statistically significant at $p \leq 0.005$, calculated with approximate randomization (Yeh, 2000).

[12] MateCat – `http://www.matecat.com/`

[13] `http://eur-lex.europa.eu/`

user_change								
Legal Domain								
Train	Test	Δ HTER	μ MAE	Batch MAE	Adaptive MAE	Adaptive Alg.	Empty MAE	Empty Alg.
rad	cons	20.5	21.4	20.6	14.5	PA	**12.5**	OSVR
cons	rad	19.4	21.2	21.3	16.1	PA	**11.3**	OSVR
sim1	sim2	3.3	14.7	**12.2**	12.6*	OSVR	12.9*	OSVR
sim2	sim1	3.2	13.4	**13.3**	13.9*	OSVR	15.2*	OSVR
IT Domain								
Train	Test	Δ HTER	μ MAE	Batch MAE	Adaptive MAE	Adaptive Alg	Empty MAE	Empty Alg
cons	rad	12.8	19.2	19.8	17.5*	OSVR	**16.6**	OSVR
rad	cons	9.6	16.8	16.6	15.6	PA	**15.5**	OSVR
sim2	sim1	3.3	14.7	**14.4**	15*	OSVR	15.5*	OSVR
sim1	sim2	1.1	15	**13.9**	14.4*	OSVR	16.1*	OSVR

Table 2: MAE of the best performing *batch*, *adaptive* and *empty* models on CAT data collected from different users in the same domain.

within the same domain (§6.1), as well as user and domain changes at the same time (§6.2).

For each document *D* (L or IT), these two scenarios are obtained by dividing *D* into two parts of equal size (80 instances for L and 140 for IT). The result is one training set and one test set for each post-editor within the same domain. For the `user_change` experiments, training and test sets are selected from different post-editors within the same domain. For the `user+domain_change` experiments, training and test sets are selected from different post-editors in different domains.

On each combination of training and test sets, the *batch*, *adaptive*, and *empty* models are trained and evaluated in terms of global MAE scores on the test set.

6.1 Dealing with user changes

Among the possible combinations of training and test data from different post-editors in the same domain, Table 2 refers to two opposite scenarios. For each domain, these respectively involve the most dissimilar and the most similar post-editors according to the ΔHTER. Also in this case, for each model (*batch*, *adaptive* and *empty*) we only report the MAE of the best performing algorithm.

The first scenario defines a challenging situation where two post-editors (*rad* and *cons*) are characterized by opposite behaviour. As evidenced by the high ΔHTER values, one of them (*rad*) is the most "radical" post-editor (performing more corrections) while the other (*cons*) is the most "conservative" one. As shown in Table 2, global MAE scores for the online algorithms (both *adaptive* and *empty*) indicate their good adaptation capabilities. This is evident from the significant improvements both over the baseline (μ) and the batch models. Interestingly, the best results are always achieved by the *empty* models (with MAE reductions up to 10 points when tested on *rad* in the L domain, and 3.2 points when tested on *rad* in the IT domain). These results (MAE reductions are always statistically significant) suggest that, when dealing with datasets with very different label distributions, the evident limitations of batch methods are more easily overcome by learning from scratch from the feedback of a new post-editor. This also holds when the amount of test points to learn from is limited, as in the L domain where the test set contains only 80 instances. From the application-oriented perspective that motivates our work, considering the high costs of acquiring large and representative QE training data, this is an important finding.

The second scenario defines a less challenging situation where the two post-editors (*sim1* and *sim2*) are characterized by the most similar behaviour (small ΔHTER). This scenario is closer to the situation described in Section §5. Also in this case MAE results for the *adaptive* and *empty* models are slightly worse, but not significantly, than those of the batch models and the baseline. However, considering the very small amount of "uninformative" instances to learn from (especially for the *empty* models), these lower results are not surprising.

A closer look at the behaviour of the online algorithms in the two domains leads to other observations. First, OSVR always outperforms PA for the *empty* models and when post-editors have sim-

user+domain_change								
Train	Test	Δ HTER	μ MAE	Batch MAE	Adaptive MAE	Adaptive Alg	Empty MAE	Empty Alg
L cons	IT rad	24.5	26.4	27	18.2	OSVR	**16.6**	OSVR
IT rad	L cons	24.0	24.9	25.4	19.7	OSVR	**12.5**	OSVR
L rad	L cons	20.5	21.4	20.6	14.5	PA	**12.5**	OSVR
L cons	L rad	19.4	21.2	21.3	16.1	PA	**11.3**	OSVR
IT cons	L cons	13.5	17.3	17.5	15.7	OSVR	**12.5**	OSVR
IT cons	IT rad	12.8	19.2	19.8	17.5	OSVR	**16.6**	OSVR
L cons	IT cons	12.7	17.6	17.6	**15.1**	OSVR	15.5	OSVR
IT rad	IT cons	9.6	16.8	16.6	15.6	PA	**15.5**	OSVR
IT cons	L rad	8.3	12.3	13	**10.7**	OSVR	11.3	OSVR
L rad	IT rad	6.8	17	16.9	**16.2**	OSVR	16.6	OSVR
L rad	IT cons	5.0	15.4	16.2	**14.7**	OSVR	15.5	OSVR
IT rad	L rad	2.2	10.6	10.8	**10.5**	OSVR	11.3	OSVR

Table 3: MAE of the best performing *batch*, *adaptive* and *empty* models on CAT data collected from different users and domains.

ilar behaviour, which are situations where the algorithm does not have to quickly adapt or react to sudden changes.

Second, PA seems to perform better for the *adaptive* models when the post-editors have significantly different behaviour and a quick adaptation to the incoming points is required. This can be motivated by the fact that PA relies on a simpler and less robust learning strategy that does not keep track of all the information coming from the previously processed instances, and can easily modify its weights taking into consideration the last seen point (see Section §3). For OSVR the addition of new points to the support set may have a limited effect on the whole model, in particular if the number of points in the set is large. This also results in a different processing time for the two algorithms.[14] For instance, in the *empty* configurations on IT data, OSVR devotes $6.0\ ms$ per instance to update the model, while PA devotes $4.8\ ms$, which comes at the cost of lower performance.

6.2 Dealing with user and domain changes

In the last round of experiments we evaluate the reactivity of different online models to simultaneous user and domain changes. To this aim, our QE models are created using a training set coming from one domain (L or IT), and then used to predict the HTER labels for the test instances coming from the other domain (*e.g.* training on L, testing on IT).

Among the possible combinations of training and test data, Table 3 refers to scenarios involving the most conservative and radical post-editors in each domain (previously identified with *cons* and *rad*)[15]. In the table, results are ordered according to the ΔHTER computed between the selected post-editor in the training domain (*e.g. L cons*) and the selected post-editor in the test domain (*e.g. IT rad*). For the sake of comparison, we also report (grey rows) the results of the experiments within the same domain presented in §6.1. For each type of model (*batch*, *adaptive* and *empty*) we only show the MAE obtained by the best performing algorithm.

Intuitively, dealing with simultaneous user and domain changes represents a more challenging problem compared to the previous setting where only post-editors changes were considered. Such intuition is confirmed by the results of the *adaptive* models that outperform both the baseline (μ) and the *batch* models even for low ΔHTER values. Although in these cases the distance between training and test data is comparable to the experiments with similar post-editors working in the same domain (*sim1* and *sim2*), here the predictive power of the *batch* models seems in fact to be lower. The same holds also for the *empty* models except in two cases where the ΔHTER is the smallest (2.2 and 5.0). This is a strong evidence of the fact that, in case of domain changes, online models can still learn from new test instances even if they have a label distribution similar to the training set.

When the distance between training and test increases, our results confirm our previous findings

[14] Their complexity depends on the number of features (*f*) and the number of previously seen instances (*n*). While for PA it is linear in *f*, *i.e. O(f)*, for OSVR it is quadratic in *n*, *i.e.* $O(n^2 * f)$.

[15] For brevity, we omit the results for the other post-editors which, however, show similar trends with respect to the previous experiments.

about the potential of the *empty* models. The observed MAE reductions range in fact from 10.4 to 12.9 points for the two combinations with the highest ΔHTER.

From the algorithmic point of view, our results indicate that OSVR achieves the best performance for all the combinations involving user and domain changes. This contrasts with the results of most of the combinations involving only user changes with post-editors characterized by opposite behaviour (grey rows in Table 3). However, it has to be remarked that in the case of heterogeneous datasets the difference between the two algorithms is always very high. In our experiments, when PA outperforms OSVR, its MAE results are significantly lower and vice-versa (respectively up to 1.5 and 1.7 MAE points). This suggests that, although PA is potentially capable of achieving higher results and better adapt to the new test points, its instability makes it less reliable for practical use.

As a final analysis of our results, we investigated how the performance of the different types of models (*batch*, *adaptive*, *empty*) relates to the distance between training and test sets. To this aim, we computed the Pearson correlation between the ΔHTER (column 3 in Table 3) and the MAE of each model (columns 5, 6 and 8), which respectively resulted in 0.9 for the *batch*, 0.63 for the *adaptive* and -0.07 for the *empty* model. These values confirm that *batch* models are heavily affected by the dissimilarity between training and test data: large differences in the label distribution imply higher MAE results and vice-versa. This is in line with our previous findings about *batch* models that, learning only from the training set, cannot leverage possible dissimilarities of the test set. The lower correlation observed for the *adaptive* models also confirms our intuitions: adapting to the new test points, these models are in fact more robust to differences with the training data. As expected, the results of the *empty* models are completely uncorrelated with the ΔHTER since they only use the test set.

This analysis confirms that, even when dealing with different domains, the similarity between the training and test data is one of the main factors that should drive the choice of the QE model. When this distance is minimal, *batch* models can be a reasonable option, but when the gap between training and test data increases, *adaptive* or *empty* models are a preferable choice to achieve good results.

7 Conclusion

In the CAT scenario, each translation job can be seen as a complex situation where the user (his personal style and background), the source document (the language and the domain) and the underlying technology (the translation memory and the MT engine that generate translation suggestions) contribute to make the task unique. So far, the adaptability to such specificities (a major challenge for CAT technology) has been mainly supported by the evolution of translation memories, which incrementally store translated segments incorporating the user style. The wide adoption of translation memories demonstrates the importance of capitalizing on such information to increase translators productivity.

While this lesson recently motivated research on adaptive MT decoders that learn from user corrections, nothing has been done to develop adaptive QE components. In the first attempt to address this problem, we proposed the application of the online learning protocol to leverage users feedback and to tailor QE predictions to their quality standards. Besides highlighting the limitations of current batch methods to adapt to user and domain changes, we performed an application-oriented analysis of different online algorithms focusing on specific aspects relevant to the CAT scenario. Our results show that the wealth of dynamic knowledge brought by user corrections can be exploited to refine in a stepwise fashion the quality judgements in different testing conditions (user changes as well as simultaneous user and domain changes).

As an additional contribution, to spark further research on this facet of the QE problem, our adaptive QE infrastructure (integrating all the components and the algorithms described in this paper) has been released as open source. Its C++ implementation is available at `http://hlt.fbk.eu/technologies/aqet`.

Acknowledgements

This work has been partially supported by the EC-funded project MateCat (ICT-2011.4.2-287688).

References

Daniel Beck, Kashif Shah, Trevor Cohn, and Lucia Specia. 2013. SHEF-Lite: When less is more for translation quality estimation. In *Proceedings of the*

8^{th} *Workshop on Statistical Machine Translation*, Sofia, Bulgaria, August.

Nicola Bertoldi, Mauro Cettolo, and Federico Marcello. 2013. Cache-based Online Adaptation for Machine Translation Enhanced Computer Assisted Translation. In *Proceedings of the XIV Machine Translation Summit*, pages 1147–1162, Nice, France.

Ergun Bicici. 2013. Feature decay algorithms for fast deployment of accurate statistical machine translation systems. In *Proceedings of the 8^{th} Workshop on Statistical Machine Translation*, Sofia, Bulgaria, August.

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. 2003. Confidence Estimation for Machine Translation. Summer workshop final report, JHU/CLSP.

Ondrej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the 8^{th} Workshop on Statistical Machine Translation*, WMT-2013, pages 1–44, Sofia, Bulgaria.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 Workshop on Statistical Machine Translation. In *Proceedings of the 7^{th} Workshop on Statistical Machine Translation (WMT'12)*, pages 10–51, Montréal, Canada.

Nicolò Cesa-Bianchi, Gabriel Reverberi, and Sandor Szedmak. 2008. Online Learning Algorithms for Computer-Assisted Translation. Deliverable D4.2, SMART: Statistical Multilingual Analysis for Retrieval and Translation.

Trevor Cohn and Lucia Specia. 2013. Modelling Annotator Bias with Multi-task Gaussian Processes: An Application to Machine Translation Quality Estimation. In *Proceedings of the 51^{st} Annual Meeting of the Association for Computational Linguistics*, ACL-2013, pages 32–42, Sofia, Bulgaria.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online Passive-Aggressive Algorithms. *J. Mach. Learn. Res.*, 7:551–585, December.

José G.C. de Souza, Christian Buck, Marco Turchi, and Matteo Negri. 2013a. FBK-UEdin participation to the WMT13 quality estimation shared task. In *Proceedings of the 8^{th} Workshop on Statistical Machine Translation*, Sofia, Bulgaria, August.

José G.C. de Souza, Miquel Esplà-Gomis, Marco Turchi, and Matteo Negri. 2013b. Exploiting Qualitative Information from Automatic Word Alignment for Cross-lingual NLP Tasks. In *Proceedings of the 51^{st} Annual Meeting of the Association for Computational Linguistics - Short Papers*, pages 771–776, Sofia, Bulgaria.

Christian Hardmeier, Joakim Nivre, and Jörg Tiedemann. 2012. Tree Kernels for Machine Translation Quality Estimation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation (WMT'12)*, pages 109–113, Montréal, Canada.

Zhengyan He and Houfeng Wang. 2012. A Comparison and Improvement of Online Learning Algorithms for Sequence Labeling. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*, pages 1147–1162, Mumbai, India.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45^{th} Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180.

Maarit Koponen, Wilker Aziz, Luciana Ramos, and Lucia Specia. 2012. Post-editing Time as a Measure of Cognitive Effort. In *Proceedings of the AMTA 2012 Workshop on Post-editing Technology and Practice (WPTP 2012)*, San Diego, California.

Maarit Koponen. 2012. Comparing Human Perceptions of Post-editing Effort with Post-editing Operations. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 181–190, Montréal, Canada.

Nick Littlestone. 1988. Learning Quickly when Irrelevant Attributes Abound: A New Linear-Threshold Algorithm. In *Machine Learning*, pages 285–318.

Junshui Ma, James Theiler, and Simon Perkins. 2003. Accurate Online Support Vector Regression. *Neural Computation*, 15:2683–2703.

Pascual Martínez-Gómez, Germán Sanchis-Trilles, and Francisco Casacuberta. 2011. Online Learning via Dynamic Reranking for Computer Assisted Translation. In *Proceedings of the 12th international conference on Computational linguistics and intelligent text processing - Volume Part II*, CICLing'11.

Pascual Martínez-Gómez, Germán Sanchis-Trilles, and Francisco Casacuberta. 2012. Online adaptation strategies for statistical machine translation in post-editing scenarios. *Pattern Recognition*, 45(9):3193–3203, September.

Prashant Mathur, Mauro Cettolo, and Marcello Federico. 2013. Online Learning Approaches in Computer Assisted Translation. In *Proceedings of the 8^{th} Workshop on Statistical Machine Translation*, Sofia, Bulgaria.

Yashar Mehdad, Matteo Negri, and Marcello Federico. 2012. Match without a Referee: Evaluating MT Adequacy without Reference Translations. In *Proceedings of the 7th Workshop on Statistical Machine Translation*, pages 171–180, Montréal, Canada.

Daniel Ortiz-Martínez, Ismael García-Varea, and Francisco Casacuberta. 2010. Online learning for interactive statistical machine translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 546–554, Stroudsburg, PA, USA.

Francesco Parrella. 2007. Online support vector regression. *Master's Thesis, Department of Information Science, University of Genoa, Italy*.

Raphael Rubino, José G.C. de Souza, Jennifer Foster, and Lucia Specia. 2013a. Topic Models for Translation Quality Estimation for Gisting Purposes. In *Proceedings of the Machine Translation Summit XIV*, Nice, France.

Raphael Rubino, Antonio Toral, S Cortés Vaíllo, Jun Xie, Xiaofeng Wu, Stephen Doherty, and Qun Liu. 2013b. The CNGL-DCU-Prompsit translation systems for WMT13. In *Proceedings of the 8th Workshop on Statistical Machine Translation*, pages 211–216, Sofia, Bulgaria.

Kashif Shah, Marco Turchi, and Lucia Specia. 2014. An Efficient and User-friendly Tool for Machine Translation Quality Estimation. In *Proceedings of the 9^{t}h International Conference on Language Resources and Evaluation*, Reykjavik, Iceland.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*, pages 223–231, Cambridge, Massachusetts, USA.

Radu Soricut, Nguyen Bach, and Ziyuan Wang. 2012. The SDL Language Weaver Systems in the WMT12 Quality Estimation Shared Task. In *Proceedings of the 7th Workshop on Statistical Machine Translation (WMT'12)*, pages 145–151, Montréal, Canada.

Lucia Specia, Nicola Cancedda, Marc Dymetman, Marco Turchi, and Nello Cristianini. 2009. Estimating the sentence-level quality of machine translation systems. In *Proceedings of the 13th Annual Conference of the European Association for Machine Translation (EAMT'09)*, pages 28–35, Barcelona, Spain.

Lucia Specia, Dhwaj Raj, and Marco Turchi. 2010. Machine Translation Evaluation versus Quality Estimation. *Machine translation*, 24(1):39–50.

Lucia Specia, Kashif Shah, José G.C. de Souza, and Trevor Cohn. 2013. QuEst - A Translation Quality Estimation Framework. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, ACL-2013, pages 79–84, Sofia, Bulgaria.

Marco Turchi and Matteo Negri. 2014. Automatic Annotation of Machine Translation Datasets with Binary Quality Judgements. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*, Reykjavik, Iceland.

Marco Turchi, Matteo Negri, and Marcello Federico. 2013. Coping with the Subjectivity of Human Judgements in MT Quality Estimation. In *Proceedings of the 8th Workshop on Statistical Machine Translation*, pages 240–251, Sofia, Bulgaria.

Alexander Yeh. 2000. More Accurate Tests for the Statistical Significance of Result Differences. In *Proceedings of the 18th conference on Computational linguistics (COLING 2000) - Volume 2*, pages 947–953, Saarbrucken, Germany.

Learning Grounded Meaning Representations with Autoencoders

Carina Silberer and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
c.silberer@ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

In this paper we address the problem of grounding distributional representations of lexical meaning. We introduce a new model which uses stacked autoencoders to learn higher-level embeddings from textual and visual input. The two modalities are encoded as vectors of attributes and are obtained automatically from text and images, respectively. We evaluate our model on its ability to simulate similarity judgments and concept categorization. On both tasks, our approach outperforms baselines and related models.

1 Introduction

Recent years have seen a surge of interest in single word vector spaces (Turney and Pantel, 2010; Collobert et al., 2011; Mikolov et al., 2013) and their successful use in many natural language applications. Examples include information retrieval (Manning et al., 2008), search query expansions (Jones et al., 2006), document classification (Sebastiani, 2002), and question answering (Yih et al., 2013). Vector spaces have been also popular in cognitive science figuring prominently in simulations of human behavior involving semantic priming, deep dyslexia, text comprehension, synonym selection, and similarity judgments (see Griffiths et al., 2007). In general, these models specify mechanisms for constructing semantic representations from text corpora based on the *distributional hypothesis* (Harris, 1970): words that appear in similar linguistic contexts are likely to have related meanings.

Word meaning, however, is also tied to the physical world. Words are *grounded* in the external environment and relate to sensorimotor experience (Regier, 1996; Landau et al., 1998; Barsalou, 2008). To account for this, new types of perceptually grounded distributional models have emerged. These models learn the meaning of words based on textual and perceptual input. The latter is approximated by feature norms elicited from humans (Andrews et al., 2009; Steyvers, 2010; Silberer and Lapata, 2012), visual information extracted automatically from images, (Feng and Lapata, 2010; Bruni et al., 2012a; Silberer et al., 2013) or a combination of both (Roller and Schulte im Walde, 2013). Despite differences in formulation, most existing models conceptualize the problem of meaning representation as one of learning from multiple views corresponding to different modalities. These models still represent words as vectors resulting from the combination of representations with different statistical properties that do not necessarily have a natural correspondence (e.g., text and images).

In this work, we introduce a model, illustrated in Figure 1, which learns grounded meaning representations by mapping words and images into a common embedding space. Our model uses stacked autoencoders (Bengio et al., 2007) to induce semantic representations integrating visual and textual information. The literature describes several successful approaches to multimodal learning using different variants of deep networks (Ngiam et al., 2011; Srivastava and Salakhutdinov, 2012) and data sources including text, images, audio, and video. Unlike most previous work, our model is defined at a finer level of granularity — it computes meaning representations for *individual* words and is unique in its use of *attributes* as a means of representing the textual and visual modalities. We follow Silberer et al. (2013) in arguing that an attribute-centric representation is expedient for several reasons.

Firstly, attributes provide a natural way of expressing salient properties of word meaning as demonstrated in norming studies (e.g., McRae et al., 2005) where humans often employ attributes when asked to describe a concept. Secondly, from

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 721–732,
Baltimore, Maryland, USA, June 23-25 2014.

a modeling perspective, attributes allow for easier integration of different modalities, since these are rendered in the same medium, namely, language. Thirdly, attributes are well-suited to describing visual phenomena (e.g., objects, scenes, actions). They allow to generalize to new instances for which there are no training examples available and to transcend category and task boundaries whilst offering a generic description of visual data (Farhadi et al., 2009).

Our model learns multimodal representations from attributes which are automatically inferred from text and images. We evaluate the embeddings it produces on two tasks, namely word similarity and categorization. In the first task, model estimates of word similarity (e.g., *gem–jewel* are similar but *glass–magician* are not) are compared against elicited similarity ratings. We performed a large-scale evaluation on a new dataset consisting of human similarity judgments for 7,576 word pairs. Unlike previous efforts such as the widely used WordSim353 collection (Finkelstein et al., 2002), our dataset contains ratings for visual and textual similarity, thus allowing to study the two modalities (and their contribution to meaning representation) together and in isolation. We also assess whether the learnt representations are appropriate for categorization, i.e., grouping a set of objects into meaningful semantic categories (e.g., *peach* and *apple* are members of FRUIT, whereas *chair* and *table* are FURNITURE). On both tasks, our model outperforms baselines and related models.

2 Related Work

The presented model has connections to several lines of work in NLP, computer vision research, and more generally multimodal learning. We review related work in these areas below.

Grounded Semantic Spaces Grounded semantic spaces are essentially distributional models augmented with perceptual information. A model akin to Latent Semantic Analysis (Landauer and Dumais, 1997) is proposed in Bruni et al. (2012b) who concatenate two independently constructed textual and visual spaces and subsequently project them onto a lower-dimensional space using Singular Value Decomposition.

Several other models have been extensions of Latent Dirichlet Allocation (Blei et al., 2003) where topic distributions are learned from words *and* other perceptual units. Feng and Lapata (2010) use visual words which they extract from a corpus of multimodal documents (i.e., BBC news articles and their associated images), whereas others (Steyvers, 2010; Andrews et al., 2009; Silberer and Lapata, 2012) use feature norms obtained in longitudinal elicitation studies (see McRae et al. (2005) for an example) as an approximation of the visual environment. More recently, topic models which combine both feature norms and visual words have also been introduced (Roller and Schulte im Walde, 2013). Drawing inspiration from the successful application of attribute classifiers in object recognition, Silberer et al. (2013) show that automatically predicted visual attributes act as substitutes for feature norms without any critical information loss.

The visual and textual modalities on which our model is trained are decoupled in that they are not derived from the same corpus (we would expect co-occurring images and text to correlate to some extent) but unified in their representation by natural language attributes. The use of stacked autoencoders to extract a shared lexical meaning representation is new to our knowledge, although, as we explain below related to a large body of work on deep learning.

Multimodal Deep Learning Our work employs deep learning (a.k.a deep networks) to project linguistic and visual information onto a unified representation that fuses the two modalities together. The goal of deep learning is to learn multiple levels of representations through a hierarchy of network architectures, where higher-level representations are expected to help define higher-level concepts.

A large body of work has focused on projecting words and images into a common space using a variety of deep learning methods ranging from deep and restricted Boltzman machines (Srivastava and Salakhutdinov, 2012; Feng et al., 2013), to autoencoders (Wu et al., 2013), and recursive neural networks (Socher et al., 2013b). Similar methods have been employed to combine other modalities such as speech and video (Ngiam et al., 2011) or images (Huang and Kingsbury, 2013). Although our model is conceptually similar to these studies (especially those applying stacked autoencoders), it differs considerably from them in at least two aspects. Firstly, most of these approaches aim to learn a shared representation between modalities

so as to infer some missing modality from others (e.g., to infer text from images and vice versa); in contrast, we aim to learn an optimal representation for each modality and their optimal combination. Secondly, our problem setting is different from the former studies, which usually deal with classification tasks and fine-tune the deep neural networks using training data with explicit class labels; in contrast we fine-tune our autoencoders using a semi-supervised criterion. That is, we use indirect supervision in the form of object classification in addition to the objective of reconstructing the attribute-centric input representation.

3 Autoencoders for Grounded Semantics

3.1 Background

Our model learns higher-level meaning representations for single words from textual and visual input in a joint fashion. We first briefly review autoencoders in Section 3.1 with emphasis on aspects relevant to our model which we then describe in Section 3.2.

Autoencoders An autoencoder is an unsupervised neural network which is trained to reconstruct a given input from its latent representation (Bengio, 2009). It consists of an encoder f_θ which maps an input vector $\mathbf{x^{(i)}}$ to a latent representation $\mathbf{y^{(i)}} = f_\theta(\mathbf{x^{(i)}}) = s(\mathbf{W}\mathbf{x^{(i)}} + \mathbf{b})$, with s being a non-linear activation function, such as a sigmoid function. A decoder $g_{\theta'}$ then aims to reconstruct input $\mathbf{x^{(i)}}$ from $\mathbf{y^{(i)}}$, i.e., $\hat{\mathbf{x}}^{(\mathbf{i})} = g_{\theta'}(\mathbf{y^{(i)}}) = s(\mathbf{W}'\mathbf{y^{(i)}} + \mathbf{b}')$. The training objective is the determination of parameters $\hat{\theta} = \{\mathbf{W}, \mathbf{b}\}$ and $\hat{\theta}' = \{\mathbf{W}', \mathbf{b}'\}$ that minimize the average reconstruction error over a set of input vectors $\{\mathbf{x^{(1)}}, ..., \mathbf{x^{(n)}}\}$:

$$\hat{\theta}, \hat{\theta}' = \arg\min_{\theta,\theta'} \frac{1}{n} \sum_{i=1}^{n} L(\mathbf{x^{(i)}}, g_{\theta'}(f_\theta(\mathbf{x^{(i)}}))), \quad (1)$$

where L is a loss function, such as cross-entropy. Parameters θ and θ' can be optimized by gradient descent methods.

Autoencoders are a means to learn representations of some input by retaining useful features in the encoding phase which help to reconstruct the input, whilst discarding useless or noisy ones. To this end, different strategies have been employed to guide parameter learning and constrain the hidden representation. Examples include imposing a bottleneck to produce an under-complete representation of the input, using sparse representations, or *denoising*.

Denoising Autoencoders The training criterion with denoising autoencoders is the reconstruction of clean input $\mathbf{x^{(i)}}$ given a corrupted version $\tilde{\mathbf{x}}^{(\mathbf{i})}$ (Vincent et al., 2010). The underlying idea is that the learned latent representation is good if the autoencoder is capable of reconstructing the actual input from its corruption. The reconstruction error for an input $\mathbf{x^{(i)}}$ with loss function L then is:

$$L(\mathbf{x^{(i)}}, g_{\theta'}(f_\theta(\tilde{\mathbf{x}}^{(\mathbf{i})}))) \quad (2)$$

One possible corruption process is *masking noise*, where the corrupted version $\tilde{\mathbf{x}}^{(\mathbf{i})}$ results from randomly setting a fraction v of $\mathbf{x^{(i)}}$ to 0.

Stacked Autoencoders Several (denoising) autoencoders can be used as building blocks to form a deep neural network (Bengio et al., 2007; Vincent et al., 2010). For that purpose, the autoencoders are pre-trained layer by layer, with the current layer being fed the latent representation of the previous autoencoder as input. Using this unsupervised pre-training procedure, initial parameters are found which approximate a good solution. Subsequently, the original input layer and hidden representations of all the autoencoders are stacked and all network parameters are fine-tuned with backpropagation.

To further optimize the parameters of the network, a supervised criterion can be imposed on top of the last hidden layer such as the minimization of a prediction error on a supervised task (Bengio, 2009). Another approach is to unfold the stacked autoencoders and fine-tune them with respect to the minimization of the global reconstruction error (Hinton and Salakhutdinov, 2006). Alternatively, a semi-supervised criterion can be used (Ranzato and Szummer, 2008; Socher et al., 2011) through combination of the unsupervised training criterion (global reconstruction) with a supervised criterion (prediction of some target given the latent representation).

3.2 Semantic Representations

To learn meaning representations of single words from textual and visual input, we employ stacked (denoising) autoencoders (SAEs). Both input modalities are vector-based representations of words, or, more precisely, the objects they refer to (e.g., *canary*, *trolley*). The vector dimensions correspond to textual and visual attributes, examples of which are shown in Table 1. We explain how these representations are obtained in more detail

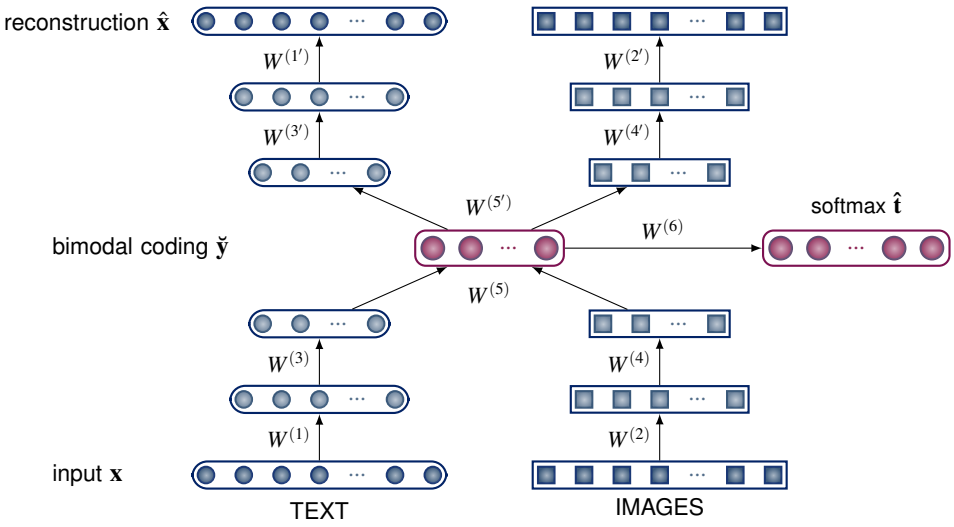


Figure 1: Stacked autoencoder trained with semi-supervised objective. Input to the model are single-word vector representations obtained from text and images. Vector dimensions correspond to textual and visual attributes, respectively (see Table 1).

in Section 4.1. We first train SAEs with two hidden layers (codings) for each modality separately. Then, we join these two SAEs by feeding their respective second coding simultaneously to another autoencoder, whose hidden layer thus yields the fused meaning representation. Finally, we stack all layers and unfold them in order to fine-tune the SAE. Figure 1 illustrates the model.

Unimodal Autoencoders For both modalities, we use the hyperbolic tangent function as activation function for encoder f_θ and decoder $g_{\theta'}$ and an entropic loss function for L. The weights of each autoencoder are tied, i.e., $\mathbf{W}' = \mathbf{W}^T$. We employ denoising autoencoders (DAEs) for pre-training the textual modality. Regarding the visual autoencoder, we derive a new ('denoised') target vector to be reconstructed for each input vector $\mathbf{x^{(i)}}$, and treat $\mathbf{x^{(i)}}$ itself as corrupted input. The unimodal autoencoder is thus trained to denoise a given input. The target vector is derived as follows: each object o in our data is represented by multiple images, and each image is in turn represented by a visual attribute vector $\mathbf{x^{(i)}}$. The target vector is the sum of $\mathbf{x^{(i)}}$ and the centroid $\mathbf{x^{(j)}}$ of the remaining attribute vectors representing object o.

Bimodal Autoencoder The bimodal autoencoder is fed with the concatenated final hidden codings of the visual and textual modalities as input and maps these inputs to a joint hidden layer $\check{\mathbf{y}}$ with B units. We normalize both unimodal input codings to unit length. Again, we use tied weights for the bimodal autoencoder. We also encourage the autoencoder to detect dependencies between the two modalities while learning the mapping to the bimodal hidden layer. We therefore apply masking noise to one modality with a masking factor v (see Section 3.1), so that the corrupted modality optimally has to rely on the other modality in order to reconstruct its missing input features.

Stacked Bimodal Autoencoder We finally build a stacked bimodal autoencoder (SAE) with all pre-trained layers and fine-tune them with respect to a semi-supervised criterion. That is, we unfold the stacked autoencoder and furthermore add a softmax output layer on top of the bimodal layer $\check{\mathbf{y}}$ that outputs predictions $\hat{\mathbf{t}}$ with respect to the inputs' object labels (e.g., *boat*):

$$\hat{\mathbf{t}}^{(\mathbf{i})} = \frac{\exp(\mathbf{W^{(6)}}\check{\mathbf{y}}^{(\mathbf{i})} + \mathbf{b^{(6)}})}{\sum_{k=1}^{O} \exp(\mathbf{W}^{(\mathbf{6})}_{\mathbf{k}.}\check{\mathbf{y}}^{(\mathbf{i})} + \mathbf{b}^{(\mathbf{6})}_{\mathbf{k}})}, \quad (3)$$

with weights $\mathbf{W^{(6)}} \in \mathbb{R}^{O\times B}$, $\mathbf{b^{(6)}} \in \mathbb{R}^{O\times 1}$, where O is the number of unique object labels. The overall objective to be minimized is therefore the weighted sum of the reconstruction error L_r and the classification error L_c:

$$L = \frac{1}{n}\sum_{i=1}^{n}\left(\delta_r L_r(\mathbf{x^{(i)}}, \hat{\mathbf{x}}^{(\mathbf{i})}) + \delta_c L_c(\mathbf{t^{(i)}}, \hat{\mathbf{t}}^{(\mathbf{i})})\right) + \lambda R \quad (4)$$

where δ_r and δ_c are weighting parameters that give different importance to the partial objectives,

		eats_seeds	has_beak	has_claws	has_handlebar	has_wheels	has_wings	is_yellow	made_of_wood
Visual	*canary*	0.05	0.24	0.15	0.00	–0.10	0.19	0.34	0.00
	trolley	0.00	0.00	0.00	0.30	0.32	0.00	0.00	0.25

		bird:n	breed:v	cage:n	chirp:v	fly:v	track:n	ride:v	run:v	rail:n	wheel:n
Textual	*canary*	0.16	0.19	0.39	0.13	0.13	0.00	0.00	0.00	0.00	–0.05
	trolley	–0.40	0.00	0.00	0.00	0.00	0.14	0.16	0.33	0.17	0.20

Table 1: Examples of attribute-based representations provided as input to our autoencoders.

L_c and L_r are entropic loss functions, and R is a regularization term with $R = \sum_{j=1}^{5} 2||\mathbf{W^{(j)}}||^2 + ||\mathbf{W^{(6)}}||^2$. Finally, $\hat{\mathbf{t}}^{(\mathbf{i})}$ is the object label vector predicted by the softmax layer for input vector $\mathbf{x}^{(\mathbf{i})}$, and $\mathbf{t}^{(\mathbf{i})}$ is the correct object label, represented as a O-dimensional one-hot vector[1].

The additional supervised criterion drives the learning towards a representation capable of discriminating between different objects. Furthermore, the semi-supervised setting affords flexibility, allowing to adapt the architecture to specific tasks. For example, by setting the corruption parameter v for the textual modality to one and δ_r to zero, a standard object classification model for images can be trained. Setting v close to one for either modality enables the model to infer the other (missing) modality. As our input consists of natural language attributes, the model would infer textual attributes given visual attributes and vice versa.

4 Experimental Setup

In this section we present our experimental setup for assessing the performance of our model. We give details on the tasks and datasets used for evaluation, we explain how the textual and visual inputs were constructed, how the SAE model was trained, and describe the approaches used for comparison with our own work.

4.1 Data

We learn meaning representations for the nouns contained in McRae et al.'s (2005) feature norms. These are 541 concrete animate and inanimate objects (e.g., animals, clothing, vehicles, utensils, fruits, and vegetables). The norms were elicited by asking participants to list properties (e.g., barks, an_animal, has_legs) describing the nouns they were presented with.

As shown in Figure 1, our model takes as input two (real-valued) vectors representing the visual and textual modalities. Vector dimensions correspond to textual and visual attributes, respectively. Textual attributes were extracted by running Strudel (Baroni et al., 2010) on a 2009 dump of the English Wikipedia.[2] Strudel is a fully automatic method for extracting weighted word-attribute pairs (e.g., *bat–species:n*, *bat–bite:v*) from a lemmatized and POS-tagged corpus. Weights are log-likelihood ratio scores expressing how strongly an attribute and a word are associated. We only retained the ten highest scored attributes for each target word. This returned a total of 2,362 dimensions for the textual vectors. Association scores were scaled to the $[-1,1]$ range.

To obtain visual vectors, we followed the methodology put forward in Silberer et al. (2013). Specifically, we used an updated version of their dataset to train SVM-based attribute classifiers that predict visual attributes for images (Farhadi et al., 2009). The dataset is a taxonomy of 636 visual attributes (e.g., has_wings, made_of_wood) and nearly 700K images from ImageNet (Deng et al., 2009) describing more than 500 of McRae et al.'s (2005) nouns. The classifiers perform reasonably well with an interpolated average precision of 0.52. We only considered attributes assigned to at least two nouns in the dataset, obtaining a 414 dimensional vector for each noun. Analogously to the textual representations, visual vectors were scaled to the $[-1,1]$ range.

We follow Silberer et al.'s (2013) partition of the dataset into training, validation, and test set and acquire visual vectors for each of the sets. We use the visual vectors of the training and development set for training the autoencoders, and the vectors for the test set for evaluation.

[1] In a one-hot vector, the element corresponding to the object label is one and the others are zero.

[2] The corpus is downloadable from `http://wacky.sslmit.unibo.it/doku.php?id=corpora`.

4.2 Model Architecture

Model parameters were optimized on a subset of the word association norms collected by Nelson et al. (1998).[3] These were established by presenting participants with a cue word (e.g., *canary*) and asking them to name an associate word in response (e.g., *bird, sing, yellow*). For each cue, the norms provide a set of associates and the frequencies with which they were named. The dataset contains a very large number of cue-associate pairs (63,619 in total) some of which luckily are covered in McRae et al. (2005).[4] During training we used correlation analysis (Spearman's ρ) to monitor the degree of linear relationship between model cue-associate (cosine) similarities and human probabilities.

The best autoencoder on the word association task obtained a correlation coefficient of 0.33. This performance is superior to the results reported in Silberer et al. (2013) (their correlation coefficients range from 0.16 to 0.28). This model has the following architecture: the textual autoencoder (see Figure 1, left-hand side) consists of 700 hidden units which are then mapped to the second hidden layer with 500 units (the corruption parameter was set to $v = 0.1$); the visual autoencoder (see Figure 1, right-hand side) has 170 and 100 hidden units, in the first and second layer, respectively. The 500 textual and 100 visual hidden units were fed to a bimodal autoencoder containing 500 latent units, and masking noise was applied to the textual modality with $v = 0.2$. The weighting parameters for the joint training objective of the stacked autoencoder were set to $\delta_r = 0.8$ and $\delta_c = 1$ (see Equation (4)).

We used the model described above and the meaning representations obtained from the output of the bimodal latent layer for all the evaluation tasks detailed below. Some performance gains could be expected if parameter optimization took place separately for each task. However, we wanted to avoid overfitting, and show that our parameters are robust across tasks and datasets.

4.3 Evaluation Tasks

Word Similarity We first evaluated how well our model predicts word similarity ratings. Although several relevant datasets exist, such as the widely used WordSim353 (Finkelstein et al., 2002) or the more recent Rel-122 norms (Szumlanski et al., 2013), they contain many abstract words, (e.g., *love–sex* or *arrest–detention*) which are not covered in McRae et al. (2005). This is for a good reason, as most abstract words do not have discernible attributes, or at least attributes that participants would agree upon. We thus created a new dataset consisting exclusively of McRae et al. (2005) nouns which we hope will be useful for the development and evaluation of grounded semantic space models.[5]

Initially, we created all possible pairings over McRae et al.'s (2005) nouns and computed their semantic relatedness using Patwardhan and Pedersen (2006)'s WordNet-based measure. We opted for this specific measure as it achieves high correlation with human ratings and has a high coverage on our nouns. Next, for each word we randomly selected 30 pairs under the assumption that they are representative of the full variation of semantic similarity. This resulted in 7,576 word pairs for which we obtained similarity ratings using Amazon Mechanical Turk (AMT). Participants were asked to rate a pair on two dimensions, visual and semantic similarity using a Likert scale of 1 (highly dissimilar) to 5 (highly similar). Each task consisted of 32 pairs covering examples of weak to very strong semantic relatedness. Two control pairs from Miller and Charles (1991) were included in each task to potentially help identify and eliminate data from participants who assigned random scores. Examples of the stimuli and mean ratings are shown in Table 2.

The elicitation study comprised overall 255 tasks, each task was completed by five volunteers. The similarity data was post-processed so as to identify and remove outliers. We considered an outlier to be any individual whose mean pairwise correlation fell outside two standard deviations from the mean correlation. 11.5% of the annotations were detected as outliers and removed. After outlier removal, we further examined how well the participants agreed in their similarity judgments. We measured inter-subject agreement as the average pairwise correlation coefficient (Spearman's ρ) between the ratings of all annotators for each task. For semantic similarity, the mean correlation was 0.76 (Min =0.34, Max

[3] http://w3.usf.edu/Freeassociation.

[4] 435 word pairs constitute the overlap between Nelson et al.'s norms (1998) and McRae et al.'s (2005) nouns.

[5] Available from http://homepages.inf.ed.ac.uk/mlap/index.php?page=resources.

Word Pairs	Semantic	Visual
football–pillow	1.0	1.2
dagger–pencil	1.0	2.2
motorcycle–wheel	2.4	1.8
orange–pumpkin	2.5	3.0
cherry–pineapple	3.6	1.2
pickle–zucchini	3.6	4.0
canary–owl	4.0	2.4
jeans–sweater	4.5	2.2
pan–pot	4.7	4.0
hornet–wasp	4.8	4.8
airplane–jet	5.0	5.0

Table 2: Mean semantic and visual similarity ratings for the McRae et al. (2005) nouns using a scale of 1 (highly dissimilar) to 5 (highly similar).

=0.97, StD =0.11) and for visual similarity 0.63 (Min =0.19, Max =0.90, SD =0.14). These results indicate that the participants found the task relatively straightforward and produced similarity ratings with a reasonable level of consistency. For comparison, Patwardhan and Pedersen's (2006) measure achieved a coefficient of 0.56 on the dataset for semantic similarity and 0.48 for visual similarity. The correlation between the average ratings of the AMT annotators and the Miller and Charles (1991) dataset was $\rho = 0.91$. In our experiments (see Section 5), we correlate model-based cosine similarities with mean similarity ratings (again using Spearman's ρ).

Categorization The task of categorization (i.e., grouping objects into meaningful categories) is a classic problem in the field of cognitive science, central to perception, learning, and the use of language. We evaluated model output against a gold standard set of categories created by Fountain and Lapata (2010). The dataset contains a classification, produced by human participants, of McRae et al.'s (2005) nouns into (possibly multiple) semantic categories (40 in total).[6]

To obtain a clustering of nouns, we used Chinese Whispers (Biemann, 2006), a randomized graph-clustering algorithm. In the categorization setting, Chinese Whispers (CW) produces a hard clustering over a weighted graph whose nodes correspond to words and edges to cosine similarity scores between vectors representing their meaning. CW is a non-parametric model, it induces the number of clusters (i.e., categories) from the data as well as which nouns belong to these clusters. In our experiments, we initialized Chinese Whispers with different graphs resulting from different vector-based representations of the McRae et al. (2005) nouns. We also transformed the dataset into hard categorizations by assigning each noun to its most typical category as extrapolated from human typicality ratings (for details see Fountain and Lapata, 2010). CW can optionally apply a minimum weight threshold which we optimized using the categorization dataset from Baroni et al. (2010). The latter contains a classification of 82 McRae et al. (2005) nouns into 10 categories. These nouns were excluded from the gold standard (Fountain and Lapata, 2010) in our final evaluation.

We evaluated the clusters produced by CW using the F-score measure introduced in the SemEval 2007 task (Agirre and Soroa, 2007); it is the harmonic mean of precision and recall defined as the number of correct members of a cluster divided by the number of items in the cluster and the number of items in the gold-standard class, respectively.

4.4 Comparison with Other Models

Throughout our experiments we compare a bimodal stacked autoencoder against unimodal autoencoders based solely on textual and visual input (left- and right-hand sides in Figure 1, respectively). We also compare our model against two approaches that differ in their fusion mechanisms. The first one is based on kernelized canonical correlation (kCCA, Hardoon et al., 2004) with a linear kernel which was the best performing model in Silberer et al. (2013). The second one emulates Bruni et al.'s (2014) fusion mechanism. Specifically, we concatenate the textual and visual vectors and project them onto a lower dimensional latent space using SVD (Golub and Reinsch, 1970). All these models run on the same datasets/items and are given input identical to our model, namely attribute-based textual and visual representations.

We furthermore report results obtained with Bruni et al.'s (2014) bimodal distributional model, which employs SVD to integrate co-occurrence-based textual representations with visual repre-

[6]The dataset can be downloaded from `http://homepages.inf.ed.ac.uk/s0897549/data/`.

	Semantic			Visual		
Models	T	V	T+V	T	V	T+V
McRae	0.71	0.49	0.68	0.58	0.52	0.62
Attributes	0.58	0.61	0.68	0.46	0.56	0.58
SAE	0.65	0.60	0.70	0.52	0.60	0.64
SVD	—	—	0.67	—	—	0.57
kCCA	—	—	0.57	—	—	0.55
Bruni	—	—	0.52	—	—	0.46
RNN-640	0.41	—	—	0.34	—	—

Table 3: Correlation of model predictions against similarity ratings for McRae et al. (2005) noun pairs (using Spearman's ρ).

#	Pair	#	Pair
1	*pliers–tongs*	11	*cello–violin*
2	*cathedral–church*	12	*cottage–house*
3	*cathedral–chapel*	13	*horse–pony*
4	*pistol–revolver*	14	*gun–rifle*
5	*chapel–church*	15	*cedar–oak*
6	*airplane–helicopter*	16	*bull–ox*
7	*dagger–sword*	17	*dress–gown*
8	*pistol–rifle*	18	*bolts–screws*
9	*cloak–robe*	19	*salmon–trout*
10	*nylons–trousers*	20	*oven–stove*

Table 4: Word pairs with highest semantic and visual similarity according to SAE model. Pairs are ranked from highest to lowest similarity.

sentations constructed from low-level image features. In their model, the textual modality is represented by the 30K-dimensional vectors extracted from UKWaC and WaCkypedia.[7] The visual modality is represented by bag-of-visual-words histograms built on the basis of clustered SIFT features (Lowe, 2004). We rebuilt their model on the ESP image dataset (von Ahn and Dabbish, 2004) using Bruni et al.'s (2013) publicly available system.

Finally, we also compare to the word embeddings obtained using Mikolov et al.'s (2011) recurrent neural network based language model. These were pre-trained on Broadcast news data (400M words) using the word2vec tool.[8] We report results with the 640-dimensional embeddings as they performed best.

5 Results

Table 3 presents our results on the word similarity task. We report correlation coefficients of model predictions against similarity ratings. As an indicator to how well automatically extracted attributes can approach the performance of clean human generated attributes, we also report results of a distributional model induced from McRae et al.'s (2005) norms (see the row labeled McRae in the table). Each noun is represented as a vector with dimensions corresponding to attributes elicited by participants of the norming study. Vector components are set to the (normalized) frequency with which participants generated the corresponding attribute. We show results for three models, using all attributes except those classified as visual (T), only visual attributes (V), and all available attributes (V+T).[9] As baselines, we also report the performance of a model based solely on textual attributes (which we obtain from Strudel), visual attributes (obtained from our classifiers), and their concatenation (see row Attributes in Table 3, and columns T, V, and T+V, respectively). The automatically obtained textual and visual attribute vectors serve as input to SVD, kCCA, and our stacked autoencoder (SAE). The third row in the table presents three variants of our model trained on textual and visual attributes only (T and V, respectively) and on both modalities jointly (T+V).

Recall that participants were asked to provide ratings on two dimensions, namely semantic and visual similarity. We would expect the textual modality to be more dominant when modeling semantic similarity and conversely the perceptual modality to be stronger with respect to visual similarity. This is borne out in our unimodal SAEs. The textual SAE correlates better with semantic similarity judgments ($\rho = 0.65$) than its visual equivalent ($\rho = 0.60$). And the visual SAE correlates better with visual similarity judgments ($\rho = 0.60$) compared to the textual SAE ($\rho = 0.52$). Interestingly, the bimodal SAE is better than the unimodal variants on both types of similarity judgments, semantic and visual. This suggests that both modalities contribute complementary information and that the SAE model is able to extract a shared representation which improves generalization performance across tasks by learning them

[7] We thank Elia Bruni for providing us with their data.

[8] Available from `http://www.rnnlm.org/`.

[9] Classification of attributes into categories is provided by McRae et al. (2005) in their dataset.

Models	T	V	T+V
McRae	0.52	0.31	0.42
Attributes	0.35	0.37	0.33
SAE	0.36	0.35	0.43
SVD	—	—	0.39
kCCA	—	—	0.37
Bruni	—	—	0.34
RNN-640	0.32	—	—

Table 5: F-score results on concept categorization.

jointly. The bimodal autoencoder (SAE, T+V) outperforms all other bimodal models on both similarity tasks. It yields a correlation coefficient of $\rho = 0.70$ on semantic similarity and $\rho = 0.64$ on visual similarity. Human agreement on the former task is 0.76 and 0.63 on the latter. Table 4 shows examples of word pairs with highest semantic and visual similarity according to the SAE model.

We also observe that simply concatenating textual and visual attributes (Attributes, T+V) performs competitively with SVD and better than kCCA. This indicates that the attribute-based representation is a powerful predictor on its own. Interestingly, both Bruni et al. (2013) and Mikolov et al. (2011) which do not make use of attributes are out-performed by all other attribute-based systems (see columns T and T+V in Table 3).

Our results on the categorization task are given in Table 5. In this task, simple concatenation of visual and textual attributes does not yield improved performance over the individual modalities (see row Attributes in Table 5). In contrast, all bimodal models (SVD, kCCA, and SAE) are better than their unimodal equivalents and RNN-640. The SAE outperforms both kCCA and SVD by a large margin delivering clustering performance similar to the McRae et al.'s (2005) norms. Table 6 shows examples of clusters produced by Chinese Whispers when using vector representations provided by the SAE model.

In sum, our experiments show that the bimodal SAE model delivers superior performance across the board when compared against competitive baselines and related models. It is interesting to note that the unimodal SAEs are in most cases better than the raw textual or visual attributes. This indicates that higher level embeddings may be beneficial to NLP tasks in general, not only to those requiring multimodal information.

STICK-LIKE UTENSILS	*baton, ladle, peg, spatula, spoon*
RELIGIOUS BUILDINGS	*cathedral, chapel, church*
WIND INSTRUMENTS	*clarinet, flute, saxophone, trombone, trumpet, tuba*
AXES	*axe, hatchet, machete, tomahawk*
FURNITURE W/ LEGS	*bed, bench, chair, couch, desk, rocker, sofa, stool, table*
FURNITURE W/O LEGS	*bookcase, bureau, cabinet, closet, cupboard, dishwasher, dresser*
LIGHTINGS	*candle, chandelier, lamp, lantern*
ENTRY POINTS	*door, elevator, gate*
UNGULATES	*bison, buffalo, bull, calf, camel, cow, donkey, elephant, goat, horse, lamb, ox, pig, pony, sheep*
BIRDS	*crow, dove, eagle, falcon, hawk, ostrich, owl, penguin, pigeon, raven, stork, vulture, woodpecker*

Table 6: Examples of clusters produced by CW using the representations obtained from the SAE model.

6 Conclusions

In this paper, we presented a model that uses stacked autoencoders to learn grounded meaning representations by simultaneously combining textual and visual modalities. The two modalities are encoded as vectors of *natural language attributes* and are obtained automatically from *decoupled* text and image data. To the best of our knowledge, our model is novel in its use of attribute-based input in a deep neural network. Experimental results in two tasks, namely simulation of word similarity and word categorization, show that our model outperforms competitive baselines and related models trained on the same attribute-based input. Our evaluation also reveals that the bimodal models are superior to their unimodal counterparts and that higher-level unimodal representations are better than the raw input. In the future, we would like to apply our model to other tasks, such as image and text retrieval (Hodosh et al., 2013; Socher et al., 2013b), zero-shot learning (Socher et al., 2013a), and word learning (Yu and Ballard, 2007).

Acknowledgment We would like to thank Vittorio Ferrari, Iain Murray and members of the ILCC at the School of Informatics for their valuable feedback. We acknowledge the support of EPSRC through project grant EP/I037415/1.

References

Agirre, Eneko and Aitor Soroa. 2007. SemEval-2007 Task 02: Evaluating Word Sense Induction and Discrimination Systems. In *Proceedings of the Fourth International Workshop on Semantic Evaluations*. Prague, Czech Republic, pages 7–12.

Andrews, M., G. Vigliocco, and D. Vinson. 2009. Integrating Experiential and Distributional Data to Learn Semantic Representations. *Psychological Review* 116(3):463–498.

Baroni, M., B. Murphy, E. Barbu, and M. Poesio. 2010. Strudel: A Corpus-Based Semantic Model Based on Properties and Types. *Cognitive Science* 34(2):222–254.

Barsalou, Lawrence W. 2008. Grounded Cognition. *Annual Review of Psychology* 59:617–845.

Bengio, Y., P. Lamblin, D. Popovici, and H. Larochelle. 2007. Greedy Layer-Wise Training of Deep Networks. In Bernhard Schölkopf, John Platt, and Thomas Hoffman, editors, *Advances in Neural Information Processing Systems 19*. MIT Press, pages 153–160.

Bengio, Yoshua. 2009. Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning* 2(1):1–127.

Biemann, Chris. 2006. Chinese Whispers – an Efficient Graph Clustering Algorithm and its Application to Natural Language Processing Problems. In *Proceedings of TextGraphs: the 1st Workshop on Graph Based Methods for Natural Language Processing*. New York, NY, pages 73–80.

Blei, D. M., A. Y. Ng, and M. I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research* 3:993–1022.

Bruni, E., G. Boleda, M. Baroni, and N. Tran. 2012a. Distributional Semantics in Technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*. Jeju Island, Korea, pages 136–145.

Bruni, E., U. Bordignon, A. Liska, J. Uijlings, and I. Sergienya. 2013. Vsem: An open library for visual semantics representation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Sofia, Bulgaria, pages 187–192.

Bruni, E., N. Tran, and M. Baroni. 2014. Multimodal distributional semantics. *J. Artif. Intell. Res. (JAIR)* 49:1–47.

Bruni, E., J. Uijlings, M. Baroni, and N. Sebe. 2012b. Distributional Semantics with Eyes: Using Image Analysis to Improve Computational Representations of Word Meaning. In *Proceedings of the 20th ACM International Conference on Multimedia*. Nara, Japan, pages 1219–1228.

Collobert, R., J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural Language Processing (almost) from Scratch. *Journal of Machine Learning Research* 12:2493–2537.

Deng, J., W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Miami, Florida, pages 248–255.

Farhadi, A., I. Endres, D. Hoiem, and D. Forsyth. 2009. Describing Objects by their Attributes. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Miami Beach, Florida, pages 1778–1785.

Feng, Fangxiang, Ruifan Li, and Xiaojie Wang. 2013. Constructing Hierarchical Image-tags Bimodal Representations for Word Tags Alternative Choice. In *Proceedings of the ICML 2013 Workshop on Challenges in Representation Learning*. Atlanta, Georgia.

Feng, Yansong and Mirella Lapata. 2010. Visual Information in Semantic Representation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Los Angeles, California, pages 91–99.

Finkelstein, L., E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. 2002. Placing Search in Context: The Concept Revisited. *ACM Transactions on Information Systems* 20(1):116–131.

Fountain, Trevor and Mirella Lapata. 2010. Meaning Representation in Natural Language Categorization. In *Proceedings of the 31st Annual Conference of the Cognitive Science Society*. Amsterdam, The Netherlands, pages 1916–1921.

Golub, Gene and Christian Reinsch. 1970. Singular Value Decomposition and Least Squares Solutions. *Numerische Mathematik* 14(5):403–420.

Griffiths, T. L., M. Steyvers, and J. B. Tenenbaum. 2007. Topics in Semantic Representation. *Psychological Review* 114(2):211–244.

Hardoon, D. R., S. R. Szedmak, and J. R. Shawe-Taylor. 2004. Canonical Correlation Analysis: An Overview with Application to Learning Methods. *Neural Computation* 16(12):2639–2664.

Harris, Zellig. 1970. Distributional Structure. In *Papers in Structural and Transformational Linguistics*, pages 775–794.

Hinton, Geoffrey E. and Ruslan R. Salakhutdinov. 2006. Reducing the Dimensionality of Data with Neural Networks. *Science* 313(5786):504–507.

Hodosh, Micah, Peter Young, and Julia Hockenmaier. 2013. Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics. *Journal of Artificial Intelligence Research* 47:853–899.

Huang, Jing and Brian Kingsbury. 2013. Audio-visual Deep Learning for Noise Robust Speech Recognition. In *Proceedings of the 38th International Conference on Acoustics, Speech, and Signal Processing*. Vancouver, Canada, pages 7596–7599.

Jones, R., B. Rey, O. Madani, and W. Greiner. 2006. Generating Query Substititions. In *Proceedings of the 15th International Conference on the World-Wide Web*. Edinburgh, Scotland, pages 387–396.

Landau, B., L. Smith, and S. Jones. 1998. Object Perception and Object Naming in Early Development. *Trends in Cognitive Science* 27:19–24.

Landauer, Thomas and Susan T. Dumais. 1997. A Solution to Plato's Problem: the Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge. *Psychological Review* 104(2):211–240.

Lowe, D. 2004. Distinctive Image Features from Scale-invariant Keypoints. *International Journal of Computer Vision* 60(2):91–110.

Manning, C. D., P. Raghavan, and H. Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY.

McRae, K., G. S. Cree, M. S. Seidenberg, and C. McNorgan. 2005. Semantic Feature Production Norms for a Large Set of Living and Nonliving Things. *Behavior Research Methods* 37(4):547–559.

Mikolov, T., S. Kombrink, L. Burget, J. Černocký, and S. Khudanpur. 2011. Extensions of Recurrent Neural Network Language Model. In *Proceedings of the 2011 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Prague, Czech Republic, pages 5528–5531.

Mikolov, T., Wen-tau Yih, and G. Zweig. 2013. Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia, pages 746–751.

Miller, George A. and Walter G. Charles. 1991. Contextual Correlates of Semantic Similarity. *Language and Cognitive Processes* 6(1).

Nelson, D. L., C. L. McEvoy, and T. A. Schreiber. 1998. The University of South Florida Word Association, Rhyme, and Word Fragment Norms.

Ngiam, Jiquan, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Ng. 2011. Multimodal Deep Learning. In *Proceedings of the 28th International Conference on Machine Learning*. Bellevue, Washington, pages 689–696.

Patwardhan, Siddharth and Ted Pedersen. 2006. Using WordNet-based Context Vectors to Estimate the Semantic Relatedness of Concepts. In *Proceedings of the EACL 2006 Workshop on Making Sense of Sense: Bringing Computational Linguistics and Psycholinguistics Together*. Trento, Italy, pages 1–8.

Ranzato, Marc'Aurelio and Martin Szummer. 2008. Semi-supervised Learning of Compact Document Representations with Deep Networks. In *Proceedings of the 25th International Conference on Machine Learning*. Helsinki, Finland, pages 792–799.

Regier, Terry. 1996. *The Human Semantic Potential*. MIT Press, Cambridge, Massachusetts.

Roller, Stephen and Sabine Schulte im Walde. 2013. A Multimodal LDA Model integrating

Textual, Cognitive and Visual Modalities. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, pages 1146–1157.

Sebastiani, Fabrizio. 2002. Machine Learning in Automated Text Categorization. *ACM Computing Surveys* 34:1–47.

Silberer, C., V. Ferrari, and M. Lapata. 2013. Models of Semantic Representation with Visual Attributes. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Sofia, Bulgaria, pages 572–582.

Silberer, Carina and Mirella Lapata. 2012. Grounded Models of Semantic Representation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Jeju Island, Korea, pages 1423–1433.

Socher, R., M. Ganjoo, C. D. Manning, and A. Y. Ng. 2013a. Zero-shot learning through cross-modal transfer. In *Advances in Neural Information Processing Systems 26*, pages 935–943.

Socher, R., Quoc V. Le, C. D. Manning, and A. Y. Ng. 2013b. Grounded Compositional Semantics for Finding and Describing Images with Sentences. In *Proceedings of the NIPS Deep Learning Workshop*.

Socher, R., J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning. 2011. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, pages 151–161.

Srivastava, Nitish and Ruslan Salakhutdinov. 2012. Multimodal Learning with Deep Boltzmann Machines. In *Advances in Neural Information Processing Systems 25*, pages 2231–2239.

Steyvers, Mark. 2010. Combining Feature Norms and Text Data with Topic Models. *Acta Psychologica* 133(3):234–342.

Szumlanski, S. R., F. Gomez, and V. K. Sims. 2013. A New Set of Norms for Semantic Relatedness Measures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Sofia, Bulgaria, pages 890–895.

Turney, Peter D. and Patrick Pantel. 2010. From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research* 37(1):141–188.

Vincent, P., H. Larochelle, I. Lajoie, Y. Bengio, and P. Manzagol. 2010. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *Journal of Machine Learning Research* 11:3371–3408.

von Ahn, Luis and Laura Dabbish. 2004. Labeling Images with a Computer Game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Vienna, Austria, pages 319–326.

Wu, Pengcheng, Steven C. H. Hoi, Hao Xia, Peilin Zhao, Dayong Wang, and Chunyan Miao. 2013. Online Multimodal Deep Similarity Learning with Application to Image Retrieval. In *Proceedings of the 21st ACM International Conference on Multimedia*. Barcelona, Spain, pages 153–162.

Yih, Wen-tau, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question Answering Using Enhanced Lexical Semantic Models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Sofia, Bulgaria, pages 1744–1753.

Yu, C. and D. H. Ballard. 2007. A Unified Model of Early Word Learning Integrating Statistical and Social Cues. *Neurocomputing* 70:2149–2165.

Joint POS Tagging and Transition-based Constituent Parsing in Chinese with Non-local Features

Zhiguo Wang
Brandeis University
Waltham, MA, USA
zgwang@brandeis.edu

Nianwen Xue
Brandeis University
Waltham, MA, USA
xuen@brandeis.edu

Abstract

We propose three improvements to address the drawbacks of state-of-the-art transition-based constituent parsers. First, to resolve the error propagation problem of the traditional pipeline approach, we incorporate POS tagging into the syntactic parsing process. Second, to alleviate the negative influence of size differences among competing action sequences, we align parser states during beam-search decoding. Third, to enhance the power of parsing models, we enlarge the feature set with non-local features and semi-supervised word cluster features. Experimental results show that these modifications improve parsing performance significantly. Evaluated on the Chinese TreeBank (CTB), our final performance reaches 86.3% (F1) when trained on CTB 5.1, and 87.1% when trained on CTB 6.0, and these results outperform all state-of-the-art parsers.

1 Introduction

Constituent parsing is one of the most fundamental tasks in Natural Language Processing (NLP). It seeks to uncover the underlying recursive phrase structure of sentences. Most of the state-of-the-art parsers are based on the PCFG paradigm and chart-based decoding algorithms (Collins, 1999; Charniak, 2000; Petrov et al., 2006). Chart-based parsers perform exhaustive search with dynamic programming, which contributes to their high accuracy, but they also suffer from higher runtime complexity and can only exploit simple local structural information.

Transition-based constituent parsing (Sagae and Lavie, 2005; Wang et al., 2006; Zhang and Clark, 2009) is an attractive alternative. It utilizes a series of deterministic shift-reduce decisions to construct syntactic trees. Therefore, it runs in linear time and can take advantage of arbitrarily complex structural features from already constructed subtrees. The downside is that they only search a tiny fraction of the whole space and are therefore commonly considered to be less accurate than chart-based parsers. Recent studies (Zhu et al., 2013; Zhang et al., 2013) show, however, that this approach can also achieve the state-of-the-art performance with improved training procedures and the use of additional source of information as features.

However, there is still room for improvement for these state-of-the-art transition-based constituent parsers. First, POS tagging is typically performed separately as a preliminary step, and POS tagging errors will propagate to the parsing process. This problem is especially severe for languages where the POS tagging accuracy is relatively low, and this is the case for Chinese where there are fewer contextual clues that can be used to inform the tagging process and some of the tagging decisions are actually influenced by the syntactic structure of the sentence. This creates a chicken and egg problem that needs to be addressed when designing a parsing model. Second, due to the existence of unary rules in constituent trees, competing candidate parses often have different number of actions, and this increases the disambiguation difficulty for the parsing model. Third, transition-based parsers have the freedom to define arbitrarily complex structural features, but this freedom has not fully been taken advantage of and most of the present approaches only use simple structural features.

In this paper, we address these drawbacks to improve the transition-based constituent parsing for Chinese. First, we integrate POS tagging into the parsing process and jointly optimize these two processes simultaneously. Because non-local syntactic information is now available to POS tag

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 733–742,
Baltimore, Maryland, USA, June 23-25 2014.

determination, the accuracy of POS tagging improves, and this will in turn improve parsing accuracy. Second, we propose a novel state alignment strategy to align candidate parses with different action sizes during beam-search decoding. With this strategy, parser states and their unary extensions are put into the same beam, therefore the parsing model could decide whether or not to use unary actions within local decision beams. Third, we take into account two groups of complex structural features that have not been previously used in transition-based parsing: *non-local* features (Charniak and Johnson, 2005) and semi-supervised *word cluster* features (Koo et al., 2008). With the help of the non-local features, our transition-based parsing system outperforms all previous single systems in Chinese. After integrating semi-supervised word cluster features, the parsing accuracy is further improved to 86.3% when trained on CTB 5.1 and 87.1% when trained on CTB 6.0, and this is the best reported performance for Chinese.

The remainder of this paper is organized as follows: Section 2 introduces the standard transition-based constituent parsing approach. Section 3 describes our three improvements to standard transition-based constituent parsing. We discuss and analyze the experimental results in Section 4. Section 5 discusses related work. Finally, we conclude this paper in Section 6.

2 Transition-based Constituent Parsing

This section describes the transition-based constituent parsing model, which is the basis of Section 3 and the baseline model in Section 4.

2.1 Transition-based Constituent Parsing Model

A transition-based constituent parsing model is a quadruple $C = (S, T, s_0, S_t)$, where S is a set of parser *states* (sometimes called *configurations*), T is a finite set of *actions*, s_0 is an initialization function to map each input sentence into a unique *initial state*, and $S_t \in S$ is a set of *terminal states*. Each action $t \in T$ is a transition function to transit a state into a new state. A parser state $s \in S$ is defined as a tuple $s = (\sigma, \beta)$, where σ is a *stack* which is maintained to hold partial subtrees that are already constructed, and β is a *queue* which is used for storing word-POS pairs that remain unprocessed. In particular, the initial state has an empty stack σ and a queue β containing the entire input sentence (word-POS pairs), and the terminal states have an empty queue β and a stack σ containing only one complete parse tree. The task of transition-based constituent parsing is to scan the input POS-tagged sentence from left to right and perform a sequence of actions to transform the initial state into a terminal state.

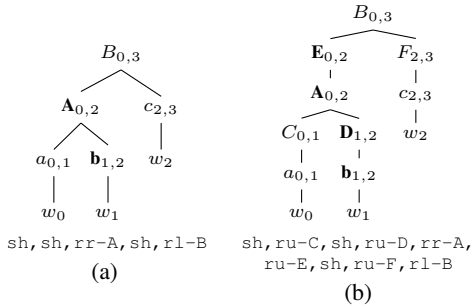


Figure 1: Two constituent trees for an example sentence $w_0w_1w_2$ with POS tags abc. The corresponding action sequences are given below, the spans of each nodes are annotated and the head nodes are written with **Bold** font type.

In order to construct lexicalized constituent parse trees, we define the following actions for the action set T according to (Sagae and Lavie, 2005; Wang et al., 2006; Zhang and Clark, 2009):

- SHIFT (`sh`): remove the first word-POS pair from β, and push it onto the top of σ;
- REDUCE-UNARY-X (`ru-x`): pop the top subtree from σ, construct a new unary node labeled with X for the subtree, then push the new subtree back onto σ. The head of the new subtree is inherited from its child;
- REDUCE-BINARY-{L/R}-X (`rl/rr-x`): pop the top two subtrees from σ, combine them into a new tree with a node labeled with X, then push the new subtree back onto σ. The left (L) and right (R) versions of the action indicate whether the head of the new subtree is inherited from its left or right child.

With these actions, our parser can process trees with unary and binary branches easily. For example, in Figure 1, for the input sentence $w_0w_1w_2$ and its POS tags abc, our parser can construct two parse trees using action sequences given below these trees. However, parse trees in Treebanks often contain an arbitrary number of branches. To

Type	Feature Templates
unigrams	p_0tc, p_0wc, p_1tc, p_1wc, p_2tc p_2wc, p_3tc, p_3wc, q_0wt, q_1wt q_2wt, q_3wt, $p_{0l}wc$, $p_{0r}wc$ $p_{0u}wc$, $p_{1l}wc$, $p_{1r}wc$, $p_{1u}wc$
bigrams	p_0wp_1w, p_0wp_1c, p_0cp_1w, p_0cp_1c p_0wq_0w, p_0wq_0t, p_0cq_0w, p_0cq_0t q_0wq_1w, q_0wq_1t, q_0tq_1w, q_0tq_1t p_1wq_0w, p_1wq_0t, p_1cq_0w, p_1cq_0t
trigrams	$p_0cp_1cp_2c$, $p_0wp_1cp_2c$, $p_0cp_1wq_0t$ $p_0cp_1cp_2w$, $p_0cp_1cq_0t$, $p_0wp_1cq_0t$ $p_0cp_1wq_0t$, $p_0cp_1cq_0w$

Table 1: Baseline features, where p_i represents the i_{th} subtree in the stack σ and q_i denotes the i_{th} item in the queue β. w refers to the head lexicon, t refers to the head POS, and c refers to the constituent label. p_{il} and p_{ir} refer to the left and right child for a binary subtree p_i, and p_{iu} refers to the child of a unary subtree p_i.

process such trees, we employ binarization and debinarization processes described in Zhang and Clark (2009) to transform multi-branch trees into binary-branch trees and restore the generated binary trees back to their original forms.

2.2 Modeling, Training and Decoding

To determine which action $t \in T$ should the parser perform at a state $s \in S$, we use a linear model to score each possible $\langle s, t \rangle$ combination:

$$score(s,t) = \vec{w} \cdot \phi(s,t) = \sum_i w_i f_i(s,t) \quad (1)$$

where $\phi(s,t)$ is the feature function used for mapping a state-action pair into a feature vector, and $\vec{w}$ is the weight vector. The score of a parser state s is the sum of the scores for all state-action pairs in the transition path from the initial state to the current state. Table 1 lists the feature templates used in our baseline parser, which is adopted from Zhang and Clark (2009). To train the weight vector $\vec{w}$, we employ the averaged perceptron algorithm with early update (Collins and Roark, 2004).

We employ the beam search decoding algorithm (Zhang and Clark, 2009) to balance the trade-off between accuracy and efficiency. Algorithm 1 gives details of the process. In the algorithm, we maintain a $beam$ (sometimes called $agenda$) to keep k best states at each step. The first $beam_0$

Algorithm 1 Beam-search Constituent Parsing

Input: A POS-tagged sentence, beam size k.
Output: A constituent parse tree.

```
1:  beam_0 ← {s_0}                          ▷ initialization
2:  i ← 0                                   ▷ step index
3:  loop
4:      P ← {}                              ▷ a priority queue
5:      while beam_i is not empty do
6:          s ← POP(beam_i)
7:          for all possible t ∈ T do
8:              s_new ← apply t to s
9:              score s_new with E.q (1)
10:             insert s_new into P
11:     beam_{i+1} ← k best states of P
12:     s_best ← best state in beam_{i+1}
13:     if s_best ∈ S_t then
14:         return s_best
15:     i ← i + 1
```

is initialized with the initial state s_0 (line 1). At step i, each of the k states in $beam_i$ is extended by applying all possible actions (line 5-10). For all newly generated states, only the k best states are preserved for $beam_{i+1}$ (line 11). The decoding process repeats until the highest scored state in $beam_{i+1}$ reaches a terminal state (line 12-14).

3 Joint POS Tagging and Parsing with Non-local Features

To address the drawbacks of the standard transition-based constituent parsing model (described in Section 1), we propose a model to jointly solve POS tagging and constituent parsing with non-local features.

3.1 Joint POS Tagging and Parsing

POS tagging is often taken as a preliminary step for transition-based constituent parsing, therefore the accuracy of POS tagging would greatly affect parsing performance. In our experiment (described in Section 4.2), parsing accuracy would decrease by 8.5% in F_1 in Chinese parsing when using automatically generated POS tags instead of gold-standard ones. To tackle this issue, we integrate POS tagging into the transition-based constituent parsing process and jointly optimize these two processes simultaneously. Inspired from Hatori et al. (2011), we modify the `sh` action by assigning a POS tag for the word when it is shifted:

- SHIFT-X (`sh-x`): remove the first word from

β, assign POS tag X to the word and push it onto the top of σ.

With such an action, POS tagging becomes a natural part of transition-based parsing. However, some feature templates in Table 1 become unavailable, because POS tags for the look-ahead words are not specified yet under the joint framework. For example, for the template q_0wt , the POS tag of the first word q_0 in the queue β is required, but it is not specified yet at the present state.

To overcome the lack of look-ahead POS tags, we borrow the concept of *delayed features* originally developed for dependency parsing (Hatori et al., 2011). Features that require look-ahead POS tags are defined as delayed features. In these features, look-ahead POS tags are taken as variables. During parsing, delayed features are extracted and passed from one state to the next state. When a `sh-x` action is performed, the look-ahead POS tag of some delayed features is specified, therefore these delayed features can be transformed into normal features (by replacing variable with the newly specified POS tag). The remaining delayed features will be transformed similarly when their look-ahead POS tags are specified during the following parsing steps.

3.2 State Alignment

Assuming an input sentence contains n words, in order to reach a terminal state, the initial state requires n `sh-x` actions to consume all words in β, and $n-1$ `rl`/`rr-x` actions to construct a complete parse tree by consuming all the subtrees in σ. However, `ru-x` is a very special action. It only constructs a new unary node for the subtree on top of σ, but does not consume any items in σ or β. As a result, the number of `ru-x` actions varies among terminal states for the same sentence. For example, the parse tree in Figure 1a contains no `ru-x` action, while the parse tree for the same input sentence in Figure 1b contains four `ru-x` actions. This makes the lengths of complete action sequences very different, and the parsing model has to disambiguate among terminal states with varying action sizes. Zhu et al. (2013) proposed a *padding* method to align terminal states containing different number of actions. The idea is to append some *IDLE* actions to terminal states with shorter action sequence, and make sure all terminal states contain the same number of actions (including *IDLE* actions).

Algorithm 2 Beam-search with State Alignment

Input: A word-segmented sentence, beam size k.
Output: A constituent parse tree.

1: $beam_0 \leftarrow \{s_0\}$ ▷ initialization
2: **for** $i \leftarrow 0$ to $2n-1$ **do** ▷ n is sentence length
3: $P_0 \leftarrow \{\}$, $P_1 \leftarrow \{\}$ ▷ two priority queues
4: **while** $beam_i$ is not empty **do**
5: $s \leftarrow$ POP($beam_i$)
6: **for** $t \in \{$`sh-x`, `rl-x`, `rr-x`$\}$ **do**
7: $s_{new} \leftarrow$ apply t to s
8: score s_{new} with E.q (1)
9: insert s_{new} into P_0
10: **for** all state s in P_0 **do**
11: **for** all possible $t \in \{$`ru-x`$\}$ **do**
12: $s_{new} \leftarrow$ apply t to s
13: score s_{new} with E.q (1)
14: insert s_{new} into P_1
15: insert all states of P_1 into P_0
16: $beam_{i+1} \leftarrow k$ best states of P_0
17: **return** the best state in $beam_{2n-1}$

We propose a novel method to align states during the parsing process instead of just aligning terminal states like Zhu et al. (2013). We classify all the actions into two groups according to whether they consume items in σ or β. `sh-x`, `rl-x`, and `rr-x` belong to *consuming* actions, and `ru-x` belongs to *non-consuming* action. Algorithm 2 gives the details of our method. It is based on the beam search decoding algorithm described in Algorithm 1. Different from Algorithm 1, Algorithm 2 is guaranteed to perform $2n-1$ parsing steps for an input sentence containing n words (line 2), and divides each parsing step into two parsing phases. In the first phase (line 4-9), each of the k states in $beam_i$ is extended by consuming actions. In the second phase (line 10-14), each of the newly generated states is further extended by non-consuming actions. Then, all these states extended by both consuming and non-consuming actions are considered together (line 15), and only the k highest-scored states are preserved for $beam_{i+1}$ (line 16). After these $2n-1$ parsing steps, the highest scored state in $beam_{2n-1}$ is returned as the final result (line 17). Figure 2 shows the states aligning process for the two trees in Figure 1. We find that our new method aligns states with their `ru-x` extensions in the same $beam$, therefore the parsing model could make decisions on whether using `ru-x` actions or not within local decision

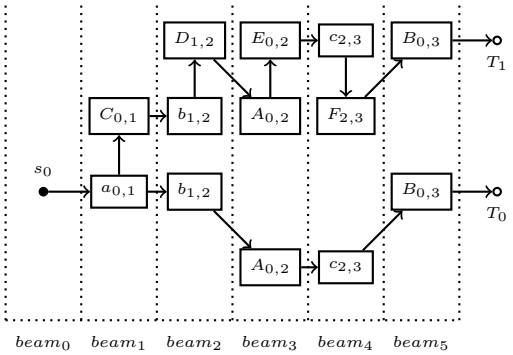


Figure 2: State alignment for the two trees in Figure 1, where s_0 is the initial state, T_0 and T_1 are terminal states corresponding to the two trees in Figure 1. For clarity, we represent each state as a rectangle with the label of top subtree in the stack σ. We also denote `sh-x` with $\rightarrow$, `ru-x` with $\uparrow$ or $\downarrow$, `rl-x` with $\nearrow$, and `rr-x` with $\searrow$.

beams.

3.3 Feature Extension

One advantage of transition-based constituent parsing is that it is capable of incorporating arbitrarily complex structural features from the already constructed subtrees in σ and unprocessed words in β. However, all the feature templates given in Table 1 are just some simple structural features. To further improve the performance of our transition-based constituent parser, we consider two group of complex structural features: *non-local* features (Charniak and Johnson, 2005; Collins and Koo, 2005) and semi-supervised *word cluster* features (Koo et al., 2008).

Table 2 lists all the non-local features we want to use. These features have been proved very helpful for constituent parsing (Charniak and Johnson, 2005; Collins and Koo, 2005). But almost all previous work considered non-local features only in parse reranking frameworks. Instead, we attempt to extract non-local features from newly constructed subtrees during the decoding process as they become incrementally available and score newly generated parser states with them. One difficulty is that the subtrees built by our baseline parser are binary trees (only the complete parse tree is debinarized into its original multi-branch form), but most of the non-local features need to be extracted from their original multi-branch forms. To resolve this conflict, we integrate the debinarization process into the parsing process, i.e., when a new subtree is constructed during parsing, we debinarize it immediately if it is not rooted with an *intermediate* node [1]. The other subtrees for subsequent parsing steps will be built based on these debinarized subtrees. After the modification, our parser can extract non-local features incrementally during the parsing process.

(Collins and Koo, 2005)	(Charniak and Johnson, 2005)	
Rules	CoPar	HeadTree
Bigrams	CoLenPar	
Grandparent Rules	RightBranch	
Grandparent Bigrams	Heavy	
Lexical Bigrams	Neighbours	
Two-level Rules	NGramTree	
Two-level Bigrams	Heads	
Trigrams	Wproj	
Head-Modifiers	Word	

Table 2: Non-local features for constituent parsing.

Semi-supervised word cluster features have been successfully applied to many NLP tasks (Miller et al., 2004; Koo et al., 2008; Zhu et al., 2013). Here, we adopt such features for our transition-based constituent parser. Given a large-scale unlabeled corpus (word segmentation should be performed), we employ the Brown cluster algorithm (Liang, 2005) to cluster all words into a binary tree. Within this binary tree, words appear as leaves, left branches are labeled with 0 and right branches are labeled with 1. Each word can be uniquely identified by its path from the root, and represented as a bit-string. By using various length of prefixes of the bit-string, we can produce word clusters of different granularities (Miller et al., 2004). Inspired from Koo et al. (2008), we employ two types of word clusters: (1) taking 4 bit-string prefixes of word clusters as replacements of POS tags, and (2) taking 8 bit-string prefixes as replacements of words. Using these two types of clusters, we construct semi-supervised word cluster features by mimicking the template structure of the original baseline features in Table 1.

4 Experiment

4.1 Experimental Setting

We conducted experiments on the Penn Chinese Treebank (CTB) version 5.1 (Xue et al., 2005): Articles 001-270 and 400-1151 were used as the training set, Articles 301-325 were used as the development set, and Articles 271-300 were used

[1] Intermediate nodes are produced by binarization process.

as the test set. Standard corpus preparation steps were performed before our experiments: empty nodes and functional tags were removed, and the unary chains were collapsed to single unary rules as Harper and Huang (2011). To build word clusters, we used the unlabeled Chinese Gigaword (LDC2003T09) and conducted Chinese word segmentation using a CRF-based segmenter.

We used EVALB [2] tool to evaluate parsing performance. The metrics include labeled precision (LP), labeled recall (LR), bracketing F_1 and POS tagging accuracy. We set the beam size k to 16, which brings a good balance between efficiency and accuracy. We tuned the optimal number of iterations of perceptron training algorithm on the development set.

4.2 Pipeline Approach *vs* Joint POS Tagging and Parsing

In this subsection, we conducted some experiments to illustrate the drawbacks of the pipeline approach and the advantages of our joint approach. We built three parsing systems: *Pipeline-Gold* system is our baseline parser (described in Section 2) taking gold-standard POS tags as input; *Pipeline* system is our baseline parser taking as input POS tags automatically assigned by Stanford POS Tagger [3]; and *JointParsing* system is our joint POS tagging and transition-based parsing system described in subsection 3.1. We trained these three systems on the training set and evaluated them on the development set. The second, third and forth rows in Table 3 show the parsing performances. We can see that the parsing F_1 decreased by about 8.5 percentage points in F_1 score when using automatically assigned POS tags instead of gold-standard ones, and this shows that the pipeline approach is greatly affected by the quality of its preliminary POS tagging step. After integrating the POS tagging step into the parsing process, our *JointParsing* system improved the POS tagging accuracy to 94.8% and parsing F_1 to 85.8%, which are significantly better than the *Pipeline* system. Therefore, the joint parsing approach is much more effective for transition-based constituent parsing.

4.3 State Alignment Evaluation

We built two new systems to verify the effectiveness of our state alignment strategy proposed in Subsection 3.2. The first system *Padding* extends our *JointParsing* system by aligning terminal states with the padding strategy proposed in Zhu et al. (2013), and the second system *StateAlign* extends the *JointParsing* system with our state alignment strategy. The fifth and sixth rows of Table 3 give the performances of these two systems. Compared with the *JointParsing* system which does not employ any alignment strategy, the *Padding* system only achieved a slight improvement on parsing F_1 score, but no improvement on POS tagging accuracy. In contrast, our *StateAlign* system achieved an improvement of 0.6% on parsing F_1 score and 0.4% on POS tagging accuracy. All these results show us that our state alignment strategy is more helpful for beam-search decoding.

System	LP	LR	F_1	POS
Pipeline-Gold	92.2	92.5	92.4	100
Pipeline	83.9	83.8	83.8	93.0
JointParsing	85.1	86.6	85.8	94.8
Padding	85.4	86.4	85.9	94.8
StateAlign	86.9	85.9	86.4	95.2
Nonlocal	88.0	86.5	87.2	95.3
Cluster	89.0	88.3	88.7	96.3
Nonlocal&Cluster	89.4	88.7	89.1	96.2

Table 3: Parsing performance on Chinese development set.

4.4 Feature Extension Evaluation

In this subsection, we examined the usefulness of the new non-local features and the semi-supervised word cluster features described in Subsection 3.3. We built three new parsing systems based on the *StateAlign* system: *Nonlocal* system extends the feature set of *StateAlign* system with non-local features, *Cluster* system extends the feature set with semi-supervised word cluster features, and *Nonlocal&Cluster* system extend the feature set with both groups of features. Parsing performances of the three systems are shown in the last three rows of Table 3. Compared with the *StateAlign* system which takes only the baseline features, the non-local features improved parsing F_1 by 0.8%, while the semi-supervised word cluster features result in an improvement of 2.3% in parsing F_1 and an 1.1% improvement on POS tagging accuracy. When integrating both groups of features, the final parsing F_1 reaches 89.1%. Al-

[2] http://nlp.cs.nyu.edu/evalb/

[3] http://nlp.stanford.edu/downloads/tagger.shtml

Type	System	LP	LR	F_1	POS
Our Systems	Pipeline	80.0	80.3	80.1	94.0
	JointParsing	82.4	83.0	82.7	95.1
	Padding	82.7	83.6	83.2	95.1
	StateAlign	84.2	82.9	83.6	95.5
	Nonlocal	85.6	84.2	84.9	95.9
	Cluster	85.2	84.5	84.9	95.8
	Nonlocal&Cluster	**86.6**	**85.9**	**86.3**	**96.0**
Single Systems	Petrov and Klein (2007)	81.9	84.8	83.3	-
	Zhu et al. (2013)	82.1	84.3	83.2	-
Reranking Systems	Charniak and Johnson (2005)*	80.8	83.8	82.3	-
	Wang and Zong (2011)	-	-	85.7	-
Semi-supervised Systems	Zhu et al. (2013)	84.4	86.8	85.6	-

Table 4: Parsing performance on Chinese test set. *Huang (2009) adapted the parse reranker to CTB5.

l these results show that both the non-local features and the semi-supervised features are helpful for our transition-based constituent parser.

4.5 Final Results on Test Set

In this subsection, we present the performances of our systems on the CTB test set. The corresponding results are listed in the top rows of Table 4. We can see that all these systems maintain a similar relative relationship as they do on the development set, which shows the stability of our systems.

To further illustrate the effectiveness of our systems, we compare them with some state-of-the-art systems. We group parsing systems into three categories: single systems, reranking systems and semi-supervised systems. Our *Pipeline*, *JointParsing*, *Padding*, *StateAlign* and *Nonlocal* systems belong to the category of single systems, because they don't utilize any extra processing steps or resources. Our *Cluster* and *Nonlocal&Cluster* systems belong to semi-supervised systems, because both of them have employed semi-supervised word cluster features. The parsing performances of state-of-the-art systems are shown in the bottom rows of Table 4. We can see that the final F_1 of our *Nonlocal* system reached 84.9%, and it outperforms state-of-the-art single systems by more than 1.6%. As far as we know, this is the best result on the CTB test set acquired by single systems. Our *Nonlocal&Cluster* system further improved the parsing F_1 to 86.3%, and it outperforms all reranking systems and semi-supervised systems. To our knowledge, this is the best reported performance in Chinese parsing.

System	F_1
Huang and Harper (2009)	85.2
Nonlocal&Cluster	**87.1**

Table 5: Parsing performance based on CTB 6.

All previous experiments were conducted on CTB 5. To check whether more labeled data can further improve our parsing system, we evaluated our *Nonlocal&Cluster* system on the Chinese TreeBank version 6.0 (CTB6), which is a super set of CTB5 and contains more annotated data. We used the same development set and test set as CTB5, and took all the remaining data as the new training set. Table 5 shows the parsing performances on CTB6. Our *Nonlocal&Cluster* system improved the final F_1 to 87.1%, which is 1.9% better than the state-of-the-art performance on CTB6 (Huang and Harper, 2009). Compared with its performance on CTB5 (in Table 4), our *Nonlocal&Cluster* system also got 0.8% improvement. All these results show that our approach can become more powerful when given more labeled training data.

4.6 Error Analysis

To better understand the linguistic behavior of our systems, we employed the berkeley-parser-analyser tool [4] (Kummerfeld et al., 2013) to categorize the errors. Table 6 presents the average

[4] http://code.google.com/p/berkeley-parser-analyser/

System	NP Int.	Unary	1-Word Span	Coord	Mod. Attach	Verb Args	Diff Label	Clause Attach	Noun Edge
Worst	1.75	0.74	0.44	0.49	0.39	0.37	0.29	0.15	0.14
Pipeline									
JointParsing									
Padding									
StateAlign									
Nonlocal									
Cluster									
Nonlocal&Cluster									
Best	1.33	0.42	0.28	0.29	0.19	0.21	0.17	0.07	0.09

Table 6: Parse errors on Chinese test set. The shaded area of each bar indicates average number of that error type per sentence, and the completely full bar indicates the number in the *Worst* row.

System	VV→NN	NN→VV	DEC→DEG	JJ→NN	NR→NN	DEG→DEC	NN→NR	NN→JJ
Worst	0.26	0.18	0.15	0.09	0.08	0.07	0.06	0.05
Pipeline								
JointParsing								
Padding								
StateAlign								
Nonlocal								
Cluster								
Nonlocal&Cluster								
Best	0.14	0.10	0.03	0.07	0.05	0.03	0.03	0.02

Table 7: POS tagging error patterns on Chinese test set. For each error pattern, the left hand side tag is the gold-standard tag, and the right hand side is the wrongly assigned tag.

number of errors for each error type by our parsing systems. We can see that almost all the *Worst* numbers are produced by the *Pipeline* system. The *JointParsing* system reduced errors of all types produced by the *Pipeline* system except for the coordination error type (Coord). The *StateAlign* system corrected a lot of the NP-internal errors (NP Int.). The *Nonlocal* system and the *Cluster* system produced similar numbers of errors for all error types. The *Nonlocal&Cluster* system produced the *Best* numbers for all the error types. NP-internal errors are still the most frequent error type in our parsing systems.

Table 7 presents the statistics of frequent POS tagging error patterns. We can see that *JointParsing* system disambiguates {VV, NN} and {DEC, DEG} better than *Pipeline* system, but cannot deal with the NN→JJ pattern very well. *StateAlign* system got better results in most of the patterns, but cannot disambiguate {NR, NN} well. *Nonlocal&Cluster* system got the best results in disambiguating the most ambiguous POS tag pairs of {VV, NN}, {DEC, DEG}, {JJ, NN} and {NN, NR}.

5 Related Work

Joint POS tagging with parsing is not a new idea. In PCFG-based parsing (Collins, 1999; Charniak, 2000; Petrov et al., 2006), POS tagging is considered as a natural step of parsing by employing lexical rules. For transition-based parsing, Hatori et al. (2011) proposed to integrate POS tagging with *dependency* parsing. Our joint approach can be seen as an adaption of Hatori et al. (2011)'s approach for *constituent* parsing. Zhang et al. (2013) proposed a transition-based constituent parser to process an input sentence from the character level. However, manual annotation of the word-internal structures need to be added to the original Treebank in order to train such a parser.

Non-local features have been successfully used for constituent parsing (Charniak and Johnson, 2005; Collins and Koo, 2005; Huang, 2008). However, almost all of the previous work use non-local features at the parse reranking stage. The reason is that the single-stage chart-based parser cannot use non-local structural features. In contrast, the transition-based parser can use arbitrarily complex structural features. Therefore, we can concisely utilize non-local features in a single-

stage parsing system.

6 Conclusion

In this paper, we proposed three improvements to transition-based constituent parsing for Chinese. First, we incorporated POS tagging into transition-based constituent parsing to resolve the error propagation problem of the pipeline approach. Second, we proposed a state alignment strategy to align competing decision sequences that have different number of actions. Finally, we enhanced our parsing model by enlarging the feature set with non-local features and semi-supervised word cluster features. Experimental results show that all these methods improved the parsing performance substantially, and the final performance of our parsing system outperformed all state-of-the-art systems.

Acknowledgments

We thank three anonymous reviewers for their cogent comments. This work is funded by the DAPRA via contract HR0011-11-C-0145 entitled "Linguistic Resources for Multilingual Processing". All opinions expressed here are those of the authors and do not necessarily reflect the views of DARPA.

References

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 173–180. Association for Computational Linguistics.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 132–139. Association for Computational Linguistics.

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 111–118, Barcelona, Spain, July.

Michael Collins. 1999. *HEAD-DRIVEN STATISTICAL MODELS FOR NATURAL LANGUAGE PARSING*. Ph.D. thesis, University of Pennsylvania.

Mary Harper and Zhongqiang Huang. 2011. Chinese statistical parsing. *Handbook of Natural Language Processing and Machine Translation*.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2011. Incremental joint pos tagging and dependency parsing in chinese. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1216–1224, Chiang Mai, Thailand, November. Asian Federation of Natural Language Processing.

Zhongqiang Huang and Mary Harper. 2009. Self-training pcfg grammars with latent annotations across languages. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 832–841. Association for Computational Linguistics.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *ACL*, pages 586–594.

Ling-Ya Huang. 2009. Improve chinese parsing with max-ent reranking parser. *Master Project Report, Brown University*.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, pages 595–603, Columbus, Ohio, June. Association for Computational Linguistics.

Jonathan K. Kummerfeld, Daniel Tse, James R. Curran, and Dan Klein. 2013. An empirical examination of challenges in chinese parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 98–103, Sofia, Bulgaria, August. Association for Computational Linguistics.

Percy Liang. 2005. *Semi-supervised learning for natural language*. Ph.D. thesis, Massachusetts Institute of Technology.

Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In *HLT-NAACL*, volume 4, pages 337–342. Citeseer.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *HLT-NAACL*, pages 404–411.

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

Kenji Sagae and Alon Lavie. 2005. A classifier-based parser with linear run-time complexity. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 125–132. Association for Computational Linguistics.

Zhiguo Wang and Chengqing Zong. 2011. Parse reranking based on higher-order lexical dependencies. In *IJCNLP*, pages 1251–1259.

Mengqiu Wang, Kenji Sagae, and Teruko Mitamura. 2006. A fast, accurate deterministic parser for chinese. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 425–432. Association for Computational Linguistics.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural language engineering*, 11(2):207–238.

Yue Zhang and Stephen Clark. 2009. Transition-based parsing of the chinese treebank using a global discriminative model. In *Proceedings of the 11th International Conference on Parsing Technologies*, pages 162–171. Association for Computational Linguistics.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2013. Chinese parsing exploiting characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 125–134, Sofia, Bulgaria, August. Association for Computational Linguistics.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 434–443, Sofia, Bulgaria, August. Association for Computational Linguistics.

Strategies for Contiguous Multiword Expression Analysis and Dependency Parsing

Marie Candito
Alpage
Paris Diderot Univ
INRIA
marie.candito@
linguist.univ-paris-diderot.fr

Matthieu Constant
Université Paris-Est
LIGM
CNRS
Matthieu.Constant@
u-pem.fr

Abstract

In this paper, we investigate various strategies to predict both syntactic dependency parsing and contiguous multiword expression (MWE) recognition, testing them on the dependency version of French Treebank (Abeillé and Barrier, 2004), as instantiated in the SPMRL Shared Task (Seddah et al., 2013). Our work focuses on using an alternative representation of syntactically regular MWEs, which captures their syntactic internal structure. We obtain a system with comparable performance to that of previous works on this dataset, but which predicts both syntactic dependencies and the internal structure of MWEs. This can be useful for capturing the various degrees of semantic compositionality of MWEs.

1 Introduction

A real-life parsing system should comprise the recognition of multi-word expressions (MWEs[1]), first because downstream semantic-oriented applications need some marking in order to distinguish between regular semantic composition and the typical semantic non-compositionality of MWEs. Second, MWE information, is intuitively supposed to help parsing.

That intuition is confirmed in a classical but non-realistic setting in which *gold* MWEs are pregrouped (Arun and Keller, 2005; Nivre and Nilsson, 2004; Eryiğit et al., 2011). But the situation is much less clear when switching to automatic MWE prediction. While Cafferkey et al. (2007) report a small improvement on the pure parsing task when using external MWE lexicons to help English parsing, Constant et al. (2012) report results on the joint MWE recognition and parsing task, in which errors in MWE recognition alleviate their positive effect on parsing performance.

While the realistic scenario of syntactic parsing with automatic MWE recognition (either done jointly or in a pipeline) has already been investigated in constituency parsing (Green et al., 2011; Constant et al., 2012; Green et al., 2013), the French dataset of the SPMRL 2013 Shared Task (Seddah et al., 2013) only recently provided the opportunity to evaluate this scenario within the framework of dependency syntax.[2] In such a scenario, a system predicts dependency trees with marked groupings of tokens into MWEs. The trees show syntactic dependencies between semantically sound units (made of one or several tokens), and are thus particularly appealing for downstream semantic-oriented applications, as dependency trees are considered to be closer to predicate-argument structures.

In this paper, we investigate various strategies for predicting from a tokenized sentence both MWEs and syntactic dependencies, using the French dataset of the SPMRL 13 Shared Task. We focus on the use of an alternative representation for those MWEs that exhibit regular internal syntax. The idea is to represent these using regular syntactic internal structure, while keeping the semantic information that they are MWEs.

We devote section 2 to related work. In section 3, we describe the French dataset, how MWEs are originally represented in it, and we present and motivate an alternative representation. Section 4 describes the different architectures we test

[1] Multiword expressions can be roughly defined as continuous or discontinuous sets of tokens, which either do not exhibit full freedom in lexical selection or whose meaning is not fully compositional. We focus in this paper on *contiguous* multiword expressions, also known as "words with spaces".

[2] The main focus of the Shared Task was on predicting both morphological and syntactic analysis for morphologically-rich languages. The French dataset is the only one containing MWEs: the French treebank has the particularity to contain a high ratio of tokens belonging to a MWE (12.7% of non numerical tokens).

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 743–753,
Baltimore, Maryland, USA, June 23-25 2014.

for predicting both syntax and MWEs. Section 5 presents the external resources targeted to improve MWE recognition. We describe experiments and discuss their results in section 6 and conclude in section 7.

2 Related work

We gave in introduction references to previous work on predicting MWEs and constituency parsing. To our knowledge, the first works[3] on *predicting* both MWEs and *dependency trees* are those presented to the SPMRL 2013 Shared Task that provided scores for French (which is the only dataset containing MWEs). Constant et al. (2013) proposed to combine pipeline and joint systems in a reparser (Sagae and Lavie, 2006), and ranked first at the Shared Task. Our contribution with respect to that work is the representation of the internal syntactic structure of MWEs, and use of MWE-specific features for the joint system. The system of Björkelund et al. (2013) ranked second on French, though with close UAS/LAS scores. It is a less language-specific system that reranks n-best dependency parses from 3 parsers, informed with features from predicted constituency trees. It uses no feature nor treatment specific to MWEs as it focuses on the general aim of the Shared Task, namely coping with prediction of morphological and syntactic analysis.

Concerning related work on the representation of MWE internal structure, we can cite the Prague Dependency Bank, which captures both regular syntax of non-compositional MWEs and their MWE status, in two distinct annotation layers (Bejček and Stranak, 2010). Our representation also resembles that of light-verb constructions (LVC) in the hungarian dependency treebank (Vincze et al., 2010): the construction has regular syntax, and a suffix is used on labels to express it is a LVC (Vincze et al., 2013).

3 Data: MWEs in Dependency Trees

The data we use is the SPMRL 13 dataset for French, in dependency format. It contains projective dependency trees that were automatically derived from the latest status of the French Treebank (Abeillé and Barrier, 2004), which consists of constituency trees for sentences from the newspaper *Le Monde*, manually annotated with phrase structures, morphological information, and grammatical functional tags for dependents of verbs. The Shared Task used an enhanced version of the constituency-to-dependency conversion of Candito et al. (2010), with different handling of MWEs. The dataset consists of 18535 sentences, split into 14759, 1235 and 2541 sentences for training, development, and final evaluation respectively.

We describe below the flat representation of MWEs in this dataset, and the modified representation for regular MWEs that we propose.

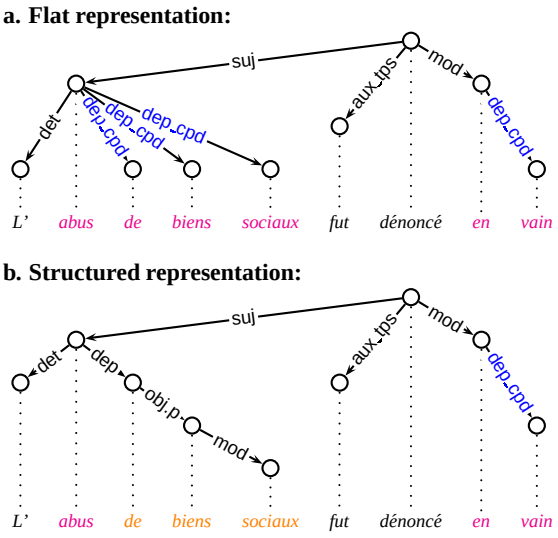


Figure 1: French dependency tree for *L'abus de biens sociaux fut dénoncé en vain* (literally *the misuse of assets social was denounced in vain*, meaning *The misuse of corporate assets was denounced in vain*), containing two MWEs (in red). Top: original flat representation. Bottom: Tree after regular MWEs structuring.

3.1 MWEs in Gold Data: Flat representation

In gold data, the MWEs appear in an expanded flat format: each MWE bears a part-of-speech and consists of a sequence of tokens (hereafter the "components" of the MWE), each having their proper POS, lemma and morphological features. In the dependency trees, there is no "node" for a MWE as a whole, but one node per MWE component (more generally one node per token). The first component of a MWE is taken as the head of the MWE. All subsequent components of the MWE depend on the first one, with the special label `dep_cpd` (hence the name *flat represen-*

[3] Concerning non contiguous MWEs, we can cite the work of Vincze et al. (2013), who experimented joint dependency parsing and light verb construction identification.

tation). Furthermore, the first MWE component bears a feature `mwehead` equal to the POS of the MWE. An example is shown in Figure 1. The MWE *en vain* (*pointlessly*) is an adverb, containing a preposition and an adjective. The latter depends on former, which bears `mwehead=ADV+`.

The algorithm to recover MWEs is: any node having dependents with the `dep_cpd` label forms a MWE with such dependents.

3.2 Alternative representation for regular MWEs

In the alternative representation we propose, irregular MWEs are unchanged and appear as flat MWEs (e.g. *en vain* in Figure 1 has pattern preposition+adjective, which is not considered regular for an adverb, and is thus unchanged). Regular MWEs appear with 'structured' syntax: we modify the tree structure to recover the regular syntactic dependencies. For instance, in the bottom tree of the figure, *biens* is attached to the preposition, and the adjective *sociaux* is attached to *biens*, with regular labels. Structured MWEs cannot be spotted using the tree topology and labels only. Features are added for that purpose: the syntactic head of the structured MWE bears a `regmwehead` for the POS of the MWE (*abus* in Figure 1), and the other components of the MWE bear a `regcomponent` feature (the orange tokens in Figure 1).[4] With this representation, the algorithm to recover regular MWEs is: any node bearing `regmwehead` forms a MWE with the set of direct or indirect dependents bearing a `regcomponent` feature.

3.2.1 Motivations

Our first motivation is to increase the quantity of information conveyed by the dependency trees, by distinguishing syntactic regularity and semantic regularity. Syntactically regular MWEs (hereafter regular MWEs) show various degrees of semantic non-compositionality. For instance, in the French Treebank, *population active* (lit. *active population*, meaning *'working population'*) is a partially compositional MWE. Furthermore, some sequences are both syntactically and semantically regular, but encoded as MWE due to frozen lexical selection. This is the case for *déficit budgétaire* (lit. *budgetary deficit*, meaning *'budget deficit'*), because it is not possible to use *déficit du budget* (*budget deficit*). Our alternative representation distinguishes between syntactic internal regularity and semantic regularity. This renders the syntactic description more uniform and it provides an internal structure for regular MWEs, which is meaningful if the MWE is fully or partially compositional. For instance, it is meaningful to have the adjective *sociaux* attach to *biens* instead of on the first component *abus*. Moreover, such a distinction opens the way to a non-binary classification of MWE status: the various criteria leading to classify a sequence as MWE could be annotated separately and using nominal or scaled categories for each criteria. For instance, *déficit budgétaire* could be marked as fully compositional, but with frozen lexical selection. Further, annotation is often incoherent for the MWEs with both regular syntax and a certain amount of semantic compositionality, the same token sequence (with same meaning) being sometimes annotated as MWE and sometimes not.

More generally, keeping a regular representation would allow to better deal with the interaction between idiomatic status and regular syntax, such as the insertion of modifiers on MWE subparts (e.g. *make a **quick** decision*).

Finally, using regular syntax for MWEs provides a more uniform training set. For instance for a sequence *N1 preposition N2*, though some *external* attachments might vary depending on whether the sequence forms a MWE or not, some may not, and the internal dependency structure $(N1 \rightarrow (preposition \rightarrow N2))$ is quite regular. One objective of the current work is to investigate whether this increased uniformity eases parsing or whether it is mitigated by the additional difficulty of finding the internal structure of a MWE.

	Total nb of MWEs	Nb of regular MWEs *(% of nouns, adverbs, prepositions, verbs)*
train	23658	12569 *(64.7, 19.2, 14.6, 1.5)*
dev	2120	1194 *(66.7, 17.7, 14.7, 0.8)*
test	4049	2051 *(64.5, 19.9, 13.6, 2.0)*

Table 1: Total number of MWEs and number of regular MWEs in training, development and test set (and broken down by POS of MWE).

[4] The syntactic head of a structured MWE may not be the first token, whereas the head token of a flat MWE is always the first one.

3.2.2 Implementation

We developed an ad hoc program for structuring the regular MWEs in gold data. MWEs are first classified as regular or irregular, using regular expressions over the sequence of parts-of-speech within the MWE. To define the regular expressions, we grouped gold MWEs according to the pair [global POS of the MWE + sequence of POS of the MWE components], and designed regular expressions to match the most frequent patterns that looked regular according to our linguistic knowledge. The internal structure for the matching MWEs was built deterministically, using heuristics favoring local attachments.[5] Table 1 shows the proportions of MWEs classified as regular, and thus further structured. About half MWEs are structured, and about two thirds of structured MWEs are nouns.

For predicted parses with structured MWEs, we use an inverse transformation of structured MWEs into flat MWEs, for evaluation against the gold data. When a predicted structured MWE is flattened, all the dependents of any token of the MWE that are not themselves belonging to the MWE are attached to the head component of the MWE.

3.3 Integration of MWE features into labels

In some experiments, we make use of alternative representations, which we refer later as "labeled representation", in which the MWE features are incorporated in the dependency labels, so that MWE composition and/or the POS of the MWE be totally contained in the tree topology and labels, and thus predictable via dependency parsing. Figure shows the labeled representation for the sentence of Figure 1.

For flat MWEs, the only missing information is the MWE part-of-speech: we concatenate it to the `dep_cpd` labels. For instance, the arc from *en* to *vain* is relabeled `dep_cpd_ADV`. For structured MWEs, in order to get full MWE account within the tree structure and labels, we need to incorporate both the MWE POS, and to mark it as

[5]The six regular expressions that we obtained cover nominal, prepositional, adverbial and verbal compounds. We manually evaluated both the regular versus irregular classification and the structuring of regular MWEs on the first 200 MWEs of the development set. 113 of these were classified as regular, and we judged that all of them were actually regular, and were correctly structured. Among the 87 classified as irregular, 7 should have been tagged as regular and structured. For 4 of them, the classification error is due to errors on the (gold) POS of the MWE components.

c. Labeled representation:

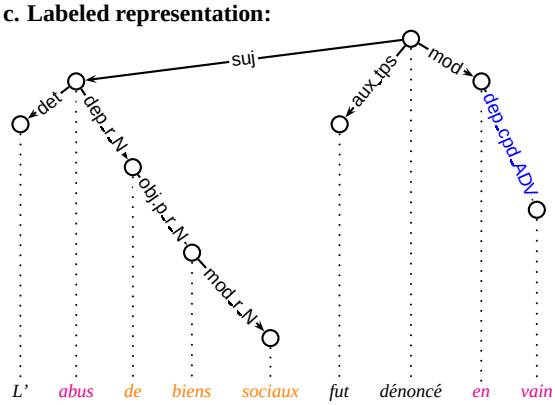


Figure 2: Integration of all MWE information into labels for the example of Figure 1.

belonging to a MWE. The suffixed label has the form `FCT_r_POS`. For instance, in bottom tree of Figure 1, arcs pointing to the non-head components (*de*, *biens*, *sociaux*) are suffixed with `_r` to mark them as belonging to a structured MWE, and with `_N` since the MWE is a noun.

In both cases, this label suffixing is translated back into features for evaluation against gold data.

4 Architectures for MWE Analysis and Parsing

The architectures we investigated vary depending on whether the MWE status of sequences of tokens is predicted via dependency parsing or via an external tool (described in section 5), and this dichotomy applies both to structured MWEs and flat MWEs. More precisely, we consider the following alternative for irregular MWEs:

- IRREG-MERGED: gold irregular MWEs are merged for training; for parsing, irregular MWEs are predicted externally, merged into one token at parsing time, and re-expanded into several tokens for evaluation;

- IRREG-BY-PARSER: the MWE status, flat topology and POS are all predicted via dependency parsing, using representations for training and parsing, with all information for irregular MWEs encoded in topology and labels (as for *in vain* in Figure 2).

For regular MWEs, their internal structure is always predicted by the parser. For instance the unlabeled dependencies for *abus de biens sociaux* are the same, independently of predicting whether it

forms a MWE or not. But we use two kinds of predictions for their MWE status and POS:

- REG-POST-ANNOTATION: the regular MWEs are encoded/predicted as shown for *abus de biens sociaux* in bottom tree of Figure 1, and their MWE status and POS is predicted after parsing, by an external tool.
- REG-BY-PARSER: all regular MWE information (topology, status, POS) is predicted via dependency parsing, using representations with all information for regular MWEs encoded in topology and labels (Figure 2).

Name	prediction of reg MWEs	prediction of irreg MWEs
JOINT	irreg-by-parser	reg-by-parser
JOINT-REG	irreg-merged	reg-by-parser
JOINT-IRREG	irreg-by-parser	reg-post-annot
PIPELINE	irreg-merged	reg-post-annot

Table 2: The four architectures, depending on how regular and irregular MWEs are predicted.

We obtain four architectures, schematized in table 2. We describe more precisely two of them, the other two being easily inferable:

JOINT-REG architecture:

- **training set**: irregular MWEs merged into one token, regular MWEs are structured, and integration of regular MWE information into the labels (`FCT_r_POS`).
- **parsing**: **(i)** MWE analysis with classification of MWEs into regular or irregular, **(ii)** merge of predicted irregular MWEs, **(iii)** tagging and morphological prediction, **(iv)** parsing

JOINT-IRREG architecture:

- **training set**: flat representation of irregular MWEs, with label suffixing (`dep_cpd_POS`), structured representation of regular MWEs without label suffixing.
- **parsing**: **(i)** MWE analysis and classification into regular or irregular, used for MWE-specific features, **(ii)** tagging and morphological prediction, **(iii)** parsing,

We compare these four architectures between them and also with two simpler architectures used by (Constant et al., 2013) within the SPMRL 13 Shared Task, in which regular and irregular MWEs are not distinguished:

Uniform joint architecture: The joint systems perform syntactic parsing and MWE analysis via a single dependency parser, using representations as in 3.3.

Uniform pipeline architecture:

- **training set**: MWEs merged into one token
- **parsing**: **(i)** MWE analysis, **(ii)** merge of predicted MWEs, **(iii)** tagging and morphological prediction, **(iv)** parsing

For each architecture, we apply the appropriate normalization procedures on the predicted parses, in order to evaluate against (i) the pseudo-gold data in structured representation, and (ii) the gold data in flat representation.

5 Use of external MWE resources

In order to better deal with MWE prediction, we use external MWE resources, namely MWE lexicons and an MWE analyzer. Both resources help to predict MWE-specific features (section 5.3) to guide the MWE-aware dependency parser. Moreover, in some of the architectures, the external MWE analyzer is used either to pre-group irregular MWEs (for the architectures using IRREG-MERGED), or to post-annotate regular MWEs.

5.1 MWE lexicons

MWE lexicons are exploited as sources of features for both the dependency parser and the external MWE analyzer. In particular, two large-coverage general-language lexicons are used: the Lefff[6] lexicon (Sagot, 2010), which contains approximately half a million inflected word forms, among which approx. $25,000$ are MWEs; and the DELA[7] (Courtois, 2009; Courtois et al., 1997) lexicon, which contains approx. one million inflected forms, among which about $110,000$ are MWEs. These resources are completed with specific lexicons freely available in the platform Unitex[8]: the toponym dictionary Prolex (Piton et al., 1999) and a dictionary of first names. Note that the lexicons do not include any information on the irregular or the regular status of the MWEs. In order to compare the MWEs present in the lexicons and those encoded in the French treebank, we applied the following procedure (hereafter called lexicon

[6] We use the version available in the POS tagger MElt (Denis and Sagot, 2009).

[7] We use the version in the platform Unitex (http://igm.univ-mlv.fr/~unitex). We had to convert the DELA POS tagset to that of the French Treebank.

[8] http://igm.univ-mlv.fr/~unitex

lookup): in a given sentence, the maximum number of non overlapping MWEs according to the lexicons are systematically marked as such. We obtain about 70% recall and 50% precision with respect to MWE spanning.

5.2 MWE Analyzer

The MWE analyzer is a CRF-based sequential labeler, which, given a tokenized text, jointly performs MWE segmentation and POS tagging (of simple tokens and of MWEs), both tasks mutually helping each other[9]. The MWE analyzer integrates, among others, features computed from the external lexicons described in section 5.1, which greatly improve POS tagging (Denis and Sagot, 2009) and MWE segmentation (Constant and Tellier, 2012). The MWE analyzer also jointly classifies its predicted MWEs as regular or irregular (the distinction being learnt on gold training set, with structured MWEs cf. section 3.2).

5.3 MWE-specific features

We introduce information from the external MWE resources in different ways:

Flat MWE features: MWE information can be integrated as features to be used by the dependency parser. We tested to incorporate the MWE-specific features as defined in the gold flat representation (section 3.1): the *mwehead=POS* feature for the MWE head token, POS being the part-of-speech of the MWE; the *component=y* feature for the non-first MWE component.

Switch: instead or on top of using the `mwehead` feature, we use the POS of the MWE instead of the POS of the first component of a flat MWE. For instance in Figure 1, the token *en* gets `pos=ADV` instead of `pos=P`. The intuition behind this feature is that for an irregular MWE, the POS of the linearly first component, which serves as head, is not always representative of the external distribution of the MWE. For regular MWEs, the usefulness of such a trick is less obvious. The first component of a regular MWE is not necessarily its head (for instance for a nominal MWE with internal pattern adjective+noun), so the switch trick could be detrimental in such cases.[10]

[9]Note that in our experiments, we use this analyzer for MWE analysis only, and discard the POS tagging prediction. Tagging is performed along with lemmatization with the Morfette tool (section 6.1).

[10]We also experimented to use POS of MWE plus suffixes to force disjoint tagsets for single words, irregular MWEs and regular MWEs, but this showed comparable results.

6 Experiments

6.1 Settings and evaluation metrics

MWE Analysis and Tagging: For the MWE analyzer, we used the tool *lgtagger*[11] (version 1.1) with its default set of feature templates, and a 10-fold jackknifing on the training corpus.

Parser: We used the second-order graph-based parser available in *Mate-tools*[12] (Bohnet, 2010). We used the *Anna3.3* version, in projective mode, with default feature sets and parameters proposed in the documentation, augmented or not with MWE-specific features, depending on the experiments.

Morphological prediction: Predicted lemmas, POS and morphology features are computed with Morfette version 0.3.5 (Chrupała et al., 2008; Seddah et al., 2010)[13], using 10 iterations for the tagging perceptron, 3 iterations for the lemmatization perceptron, default beam size for the decoding of the joint prediction, and the Lefff (Sagot, 2010) as external lexicon used for out-of-vocabulary words. We performed a 10-fold jackknifing on the training corpus.

Evaluation metrics: we evaluate our parsing systems by using the standard metrics for dependency parsing: Labeled Attachment Score (LAS) and Unlabeled Attachment Score (UAS), computed using all tokens including punctuation. To evaluate statistical significance of parsing performance differences, we use eval07.pl[14] with -b option, and then Dan Bikel's comparator.[15] For MWEs, we use the Fmeasure for recognition of untagged MWEs (hereafter FUM) and for recognition of tagged MWEs (hereafter FTM).

6.2 MWE-specific feature prediction

In all our experiments, for the switch trick (section 5.3), the POS of MWE is always predicted using the MWE analyzer. For the flat MWE features, we experimented both with features predicted by the MWE analyzer, and with features predicted using the external lexicons mentioned in section 5.1 (using the lexicon lookup procedure). Both kinds of

[11]http://igm.univ-mlv.fr/~mconstan

[12]http://code.google.com/p/mate-tools/

[13]https://sites.google.com/site/morfetteweb/

[14]http://nextens.uvt.nl/depparse-wiki/SoftwarePage

[15]The compare.pl script, formerly available at www.cis.upenn.edu/ dbikel/

					LABELED REPRES.		STRUCTURED REPRESENTATION			FLAT REPRESENTATION			
ARCHI		MWE feats	swi. irreg	swi. reg	LAS	UAS	LAS	FUM irreg	FTM irreg	LAS	UAS	FUM	FTM
JOINT	bsline	-	-	-	84.5	89.3	87.0	83.6	80.6	84.2	88.1	73.5	70.7
	best	+	+	+	**85.3**	89.7	**87.5**	**85.4**	**82.6**	**85.2**	**88.8**	77.6	74.5
JOINT-IRREG	bsline	-	-	-	84.7	89.4	87.0	83.5	80.3	84.5	88.0	78.3	75.9
	best	+	+	+	85.1	**89.8**	87.4	85.0	81.6	84.9	88.3	**79.0**	**76.5**
JOINT-REG	bsline	-	NA	-	84.2	89.1	86.7	84.2	80.8	84.0	88.0	73.3	70.3
	best	+	NA	+	84.7	89.3	86.9	84.1	80.7	84.6	88.3	76.3	73.2
PIPE LINE	bsline	-	NA	-	84.6	89.2	86.9	84.1	80.7	84.5	87.9	78.8	76.3
	best	-	NA	+	84.7	89.4	87.0	84.2	80.8	84.6	88.1	78.8	76.3

Table 3: Baseline and best results for the four MWE+parsing architectures on the dev set (see text for statistical significance evaluation). The UAS for the structured representation is the same as the one for the labeled representation, and is not repeated.

prediction lead to fairly comparable results, so in all the following, the MWE features, when used, are predicted using the external lexicons.

6.3 Tuning features for each architecture

We ran experiments for all value combinations of the following parameters: (i) the architecture, (ii) whether MWE features are used, whether the switch trick is applied or not (iii) for irregular MWEs and (iv) for regular MWEs.

We performed evaluation of the predicted parses using the three representations described in section 3, namely flat, structured and labeled representations. In the last two cases, the evaluation is performed against an instance of the gold data automatically transformed to match the representation type. Moreover, for the "labeled representation" evaluation, though the MWE information in the predicted parses is obtained in various ways, depending on the architecture, we always map all this information in the dependency labels, to obtain predicted parses matching the "labeled representation". While the evaluation in flat representation is the only one comparable to other works on this dataset, the other two evaluations provide useful information. In the "labeled representation" evaluation, the UAS provides a measure of syntactic attachments for sequences of words, independently of the (regular) MWE status of subsequences. For the sequence *abus de biens sociaux*, suppose that the correct internal structure is predicted, but not the MWE status. The UAS for labeled representation will be maximal, whereas for the flat representation, the last two tokens will count as incorrect for UAS. For LAS, in both cases the three last tokens will count as incorrect if the wrong MWE status is predicted. So to sum up on the "labeled evaluation", we obtain a LAS evaluation for the whole task of parsing plus MWE recognition, but an UAS evaluation that penalizes less errors on MWE status, while keeping a representation that is richer: predicted parses contain not only the syntactic dependencies and MWE information, but also a classification of MWEs into regular and irregular, and the internal syntactic structure of regular MWEs.

The evaluation on "structured representation" can be interpreted as an evaluation of the parsing task plus the recognition of irregular MWEs only: both LAS and UAS are measured independently of errors on regular MWE status (note the UAS is exactly the same than in the "labeled" case).

For each architecture, Table 3 shows the results for two systems: first the baseline system without any MWE features nor switches and immediately below the best settings for the architecture. The JOINT baseline corresponds to a "pure" joint system without external MWE resources (hence the minus sign for the first three columns). For each architecture except the PIPELINE one, differences between the baseline and the best setting are statistically significant ($p < 0.01$). Differences between best PIPELINE and best JOINT-REG are not. Best JOINT has statistically significant difference ($p < 0.01$) over both best JOINT-REG and best PIPELINE. The situation for best JOINT-IRREG with respect to the other three is borderline (with various p-values depending on the metrics).

Concerning the tuning of parameters, it appears that the best setting is to use MWE-features, and switch for both regular and irregular MWEs, except for the pipeline architecture for which results without MWE features are slightly better. So overall, informing the parser with independently pre-

	LABELED REPRESENTATION		STRUCTURED REPRESENTATION				FLAT REPRESENTATION			
SYSTEM	LAS	UAS	LAS	UAS	FUM irreg	FTM irreg	LAS	UAS	FUM	FTM
baseline JOINT	84.13	88.93	86.62	88.93	83.6	79.2	83.97	87.80	73.9	70.5
best JOINT	84.59	89.21	86.92	89.21	85.7	81.4	84.48	88.13	77.0	73.5
best JOINT-IRREG	84.50	89.21	86.97	89.24	86.3	82.1	84.36	87.75	78.6	75.4
best JOINT-REG	84.31	89.0	86.63	89.00	84.5	80.4	84.18	87.95	76.4	73.3
best PIPELINE	84.02	88.83	86.49	88.83	84.4	80.4	83.88	87.33	77.6	74.4

Table 4: Final results on test set for baseline and the best system for each architecture.

dicted POS of MWE has positive impact. The best architectures are JOINT and JOINT-IRREG, with the former slightly better than the latter for parsing metrics, though only some of the differences are significant between the two. It can be noted though, that JOINT-IRREG performs overall better on MWEs (last two columns of table 3), whereas JOINT performs better on irregular MWEs: the latter seems to be beneficial for parsing, but is less efficient to correctly spot the regular MWEs.

Concerning the three distinct representations, evaluating on structured representation (hence without looking at regular MWE status) leads to a rough 2 point performance increase for the LAS and a one point increase for the UAS, with respect to the evaluation against flat representation. This quantifies the additional difficulty of deciding for a regular sequence of tokens whether it forms a MWE or not. The evaluation on the labeled representation provides an evaluation of the full task (parsing, regular/irregular MWE recognition and regular MWEs structuring), with a UAS that is less impacted by errors on regular MWE status, while LAS reflects the full difficulty of the task.[16]

6.4 Results on test set and comparison

We provide the final results on the test set in table 4. We compare the baseline JOINT system with the best system for all four reg/irreg architectures (cf. section 6.3). We observe the same general trend as in the development corpus, but with tinier differences. JOINT and JOINT-IRREG significantly outperform the baseline and the PIPELINE, on labeled representation and flat representation. We can see that there is no significant difference between JOINT and JOINT-IRREG and between JOINT-REG and JOINT-IRREG. JOINT slightly outperforms JOINT-REG ($p < 0.05$). On the structured representation, the two best systems (JOINT and JOINT-IRREG) significantly outperform the other systems ($p < 0.01$ for all; $p < 0.05$ for JOINT-REG).

[16]The slight differences in LAS between the labeled and the flat representations are due to side effects of errors on MWE status: some wrong reattachments performed to obtain flat representation decrease the UAS, but also in some cases the LAS.

	DEV		TEST	
System	UAS	LAS	UAS	LAS
reg/irreg joint	88.79	85.15	88.13	84.48
Bjork13	*88.30*	*84.84*	*87.87*	*84.37*
Const13 pipeline	88.73	85.28	**88.35**	**84.91**
Const13 joint	88.21	84.60	87.76	84.14
uniform joint	**88.81**	**85.42**	87.96	84.59

Table 5: Comparison on dev set of our best architecture with reg/irregular MWE distinction (first row), with the single-parser architectures of (Constant et al., 2013) (Const13) and (Björkelund et al., 2013) (Bjork13). Uniform joint is our reimplementation of Const13 joint, enhanced with mwe-features and switch.

Moreover, we provide in table 5 a comparison of our best architecture with reg/irregular MWE distinction with other architectures that do not make this distinction, namely the two best comparable systems designed for the SPMRL Shared Task (Seddah et al., 2013): the pipeline simple parser based on Mate-tools of Constant et al. (2013) (Const13) and the Mate-tools system (without reranker) of Björkelund et al. (2013) (Bjork13). We also reimplemented and improved the uniform joint architecture of Constant et al. (2013), by adding MWE features and switch. Results can only be compared on the flat representation, because the other systems output poorer linguistic information. We computed statistical significance of differences between our systems and Const13. On dev, the best system is the enhanced uniform joint, but differences are not significant between that and the best reg/irreg joint (1st row) and the Const13 pipeline. But on the test corpus (which is twice bigger), the best system is Const13

	Tasks		LAS	UAS	ALL MWE		REG MWE		IRREG MWE	
System	Parsing	MWE			FUM	FTM	FUM	FTM	FUM	FTM
Our best system (best JOINT)	+	all	85.15	88.78	77.6	74.5	70.8	67.8	85.4	82.6
Uniform pipeline/gold MWEs	+	-	88.73	90.60	-	-	-	-	-	-
CRF-based MWE analyzer	-	all	-	-	78.8	76.3	73.5	71.9	84.2	80.8
JOINT-REG	+	all	84.58	88.34	76.3	73.2	69.3	66.5	84.1	80.7
JOINT-REG/gold irreg. MWE	+	reg.	85.86	89.19	82.9	78.8	70.0	67.2	-	-

Table 6: Comparison with simpler tasks on the flat representation of the development set.

pipeline, with statistically significant differences over our joint systems. So the first observation is that our architectures that distinguish between reg/irreg MWEs do not outperform uniform architectures. But we note that the differences are slight, and the output we obtain is enhanced with regular MWE internal structure. It can thus be noted that the increased syntactic uniformity obtained by our MWE representation is mitigated so far by the additional complexity of the task. The second observation is that currently the best system on this dataset is a pipeline system, as results on test set show (and somehow contrary to results on dev set). The joint systems that integrate MWE information in the labels seem to suffer from increased data sparseness.

6.5 Evaluating the double task with respect to simpler tasks

In this section, we propose to better evaluate the difficulty of combining the tasks of MWE analysis and dependency parsing by comparing our systems with systems performing simpler tasks: i.e. MWE recognition without parsing, and parsing with no or limited MWE recognition, simulated by using gold MWEs. We also provide a finer evaluation of the MWE recognition task, in particular with respect to their regular/irregular status.

We first compare our best system with a parser where all MWEs have been perfectly pre-grouped, in order to quantify the difficulty that MWEs add to the parsing task. We also compare the performance on MWEs of our best system with that achieved by the CRF-based analyzer described in section 5.2. Next, we compare the best JOINT-REG system with the one based on the same architecture but where the irregular MWEs are perfectly pre-identified, in order to quantify the difficulty added by the irregular MWEs. Results are given in table 6. Without any surprise, the task is much easier without considering MWE recognition. We can see that without considering MWE analysis the parsing accuracy is about 2.5 points better in terms of LAS. In the JOINT-REG architecture, assuming gold irregular MWE identification, increases LAS by 1.3 point. In terms of MWE recognition, as compared with the CRF-based analyzer, our best system is around 2 points below. But the situation is quite different when breaking the evaluation by MWE type. Our system is 1 point better than the CRF-based analyzer for irregular MWEs. This shows that considering a larger syntactic context helps recognition of irregular MWEs. The ”weak point” of our system is therefore the identification of regular MWEs.

7 Conclusion

We experimented strategies to predict both MWE analysis and dependency structure, and tested them on the dependency version of French Treebank (Abeillé and Barrier, 2004), as instantiated in the SPMRL Shared Task (Seddah et al., 2013). Our work focused on using an alternative representation of syntactically regular MWEs, which captures their syntactic internal structure. We obtain a system with comparable performance to that of previous works on this dataset, but which predicts both syntactic dependencies and the internal structure of MWEs. This can be useful for capturing the various degrees of semantic compositionality of MWEs. The main weakness of our system comes from the identification of regular MWEs, a property which is highly lexical. Our current use of external lexicons does not seem to suffice, and the use of data-driven external information to better cope with this identification can be envisaged.

References

Anne Abeillé and Nicolas Barrier. 2004. Enriching a french treebank. In *Proceedings of LREC 2004*, Lisbon, Portugal.

Abhishek Arun and Frank Keller. 2005. Lexicalization in crosslinguistic probabilistic parsing: The case of

french. In *Proceedings of ACL 2005*, Ann Arbor, USA.

Eduard Bejček and Pavel Stranak. 2010. Annotation of multiword expressions in the prague dependency treebank. *Language Resources and Evaluation*, 44:7–21.

Anders Björkelund, Özlem Çetinoğlu, Thomas Farkas, Richárdand Müller, and Wolfgang Seeker. 2013. (re)ranking meets morphosyntax: State-of-the-art results from the spmrl 2013 shared task. In *Proceedings of the 4th Workshop on Statistical Parsing of Morphologically Rich Languages: Shared Task*, Seattle, WA.

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of COLING 2010*, Beijing, China.

Conor Cafferkey, Deirdre Hogan, and Josef van Genabith. 2007. Multi-word units in treebank-based probabilistic parsing and generation. In *Proceedings of the 10th International Conference on Recent Advances in Natural Language Processing (RANLP'07)*, Borovets, Bulgaria.

Marie Candito, Benoit Crabbé, and Pascal Denis. 2010. Statistical french dependency parsing : Treebank conversion and first results. In *Proceedings of LREC 2010*, Valletta, Malta.

Grzegorz Chrupała, Georgiana Dinu, and Josef van Genabith. 2008. Learning morphology with morfette. In *Proceedings of LREC 2008*, Marrakech, Morocco. ELDA/ELRA.

Matthieu Constant and Isabelle Tellier. 2012. Evaluating the impact of external lexical resources into a crf-based multiword segmenter and part-of-speech tagger. In *Proceedings of LREC 2012*, Istanbul, Turkey.

Matthieu Constant, Anthony Sigogne, and Patrick Watrin. 2012. Discriminative strategies to integrate multiword expression recognition and parsing. In *Proceedings of ACL 2012*, Stroudsburg, PA, USA.

Matthieu Constant, Marie Candito, and Djamé Seddah. 2013. The ligm-alpage architecture for the spmrl 2013 shared task: Multiword expression analysis and dependency parsing. In *Proceedings of the 4th Workshop on Statistical Parsing of Morphologically Rich Languages: Shared Task*, Seattle, WA.

Blandine Courtois, Mylène Garrigues, Gaston Gross, Maurice Gross, René Jung, Mathieu-Colas Michel, Anne Monceaux, Anne Poncet-Montange, Max Silberztein, and Robert Vivés. 1997. Dictionnaire électronique DELAC : les mots composés binaires. Technical Report 56, University Paris 7, LADL.

Blandine Courtois. 2009. Un système de dictionnaires électroniques pour les mots simples du français. *Langue Française*, 87:11–22.

Pascal Denis and Benoît Sagot. 2009. Coupling an annotated corpus and a morphosyntactic lexicon for state-of-the-art POS tagging with less human effort. In *Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation (PACLIC'09)*, Hong Kong.

Gülşen Eryiğit, Tugay Ilbay, and Ozan Arkan Can. 2011. Multiword expressions in statistical dependency parsing. In *Proceedings of the IWPT Workshop on Statistical Parsing of Morphologically-Rich Languages (SPMRL'11)*, Dublin, Ireland.

Spence Green, Marie-Catherine de Marneffe, John Bauer, and Christofer D. Manning. 2011. Multiword expression identification with tree substitution grammars: A parsing tour de force with french. In *Proceedings of EMNLP 2011*, Edinburgh, Scotland.

Spence Green, Marie-Catherine de Marneffe, and Christopher D Manning. 2013. Parsing models for identifying multiword expressions. *Computational Linguistics*, 39(1):195–227.

Joakim Nivre and Jens Nilsson. 2004. Multiword units in syntactic parsing. In *Proceedings of the LREC Workshop : Methodologies and Evaluation of Multiword Units in Real-World Applications (MEMURA)*, Lisbon, Portugal.

Odile Piton, Denis Maurel, and Claude Belleil. 1999. The prolex data base : Toponyms and gentiles for nlp. In *Proceedings of the Third International Workshop on Applications of Natural Language to Data Bases (NLDB'99)*, Klagenfurt, Austria.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of NAACL/HLT 2006, Companion Volume: Short Papers*, Stroudsburg, PA, USA.

Benoît Sagot. 2010. The lefff, a freely available, accurate and large-coverage lexicon for french. In *Proceedings of LREC 2010*, Valletta, Malta.

Djamé Seddah, Grzegorz Chrupała, Ozlem Cetinoglu, Josef van Genabith, and Marie Candito. 2010. Lemmatization and statistical lexicalized parsing of morphologically-rich languages. In *Proceedings of the NAACL/HLT Workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL 2010)*, Los Angeles, CA.

Djamé Seddah, Reut Tsarfaty, Sandra K'ubler, Marie Candito, Jinho Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiorkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clérgerie. 2013. Overview of the spmrl 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the 4th Workshop on Statistical Parsing of Morphologically Rich Languages: Shared Task*, Seattle, WA.

Veronika Vincze, Dóra Szauter, Attila Almási, György Móra, Zoltán Alexin, and János Csirik. 2010. Hungarian dependency treebank. In *Proceedings of LREC 2010*, Valletta, Malta.

Veronika Vincze, János Zsibrita, and Istvàn Nagy T. 2013. Dependency parsing for identifying hungarian light verb constructions. In *Proceedings of International Joint Conference on Natural Language Processing (IJCNLP 2013)*, Nagoya, Japan.

Correcting Preposition Errors in Learner English Using Error Case Frames and Feedback Messages

Ryo Nagata[1*] **Mikko Vilenius**[2] **Edward Whittaker**[3]
[1]Konan University / Kobe, Japan
[2]The Japan Institute for Educational Measurement, Inc. / Tokyo, Japan
[3]Inferret Limited / Northampton, England
nagata-acl@hyogo-u.ac.jp.

Abstract

This paper presents a novel framework called *error case frames* for correcting preposition errors. They are case frames specially designed for describing and correcting preposition errors. Their most distinct advantage is that they can correct errors with feedback messages explaining why the preposition is erroneous. This paper proposes a method for automatically generating them by comparing learner and native corpora. Experiments show (i) automatically generated error case frames achieve a performance comparable to conventional methods; (ii) error case frames are intuitively interpretable and manually modifiable to improve them; (iii) feedback messages provided by error case frames are effective in language learning assistance. Considering these advantages and the fact that it has been difficult to provide feedback messages by automatically generated rules, error case frames will likely be one of the major approaches for preposition error correction.

1 Introduction

This paper presents a novel framework for correcting preposition errors. Its most significant advantage over previous methods is that it can provide learners with feedback messages, that is, explanatory notes describing why the detected preposition is erroneous and should be corrected as indicated, as shown in Fig. 1. Despite the fact that appropriate feedback messages are essential in language learning assistance (Ferris and Roberts, 2001; Robb et al., 1986), which is one of the immediate applications of grammatical error correction, almost all previous methods are incapable of providing feedback messages.

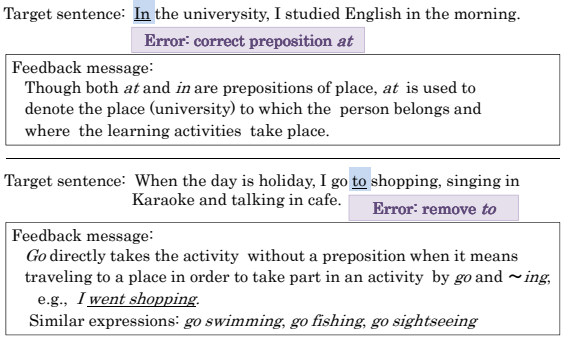


Figure 1: Error correction and feedback messages provided by the proposed method.

Grammatical error correction has been intensively studied in recent years. Current methods mostly exploit machine learning-based classifiers to correct target errors; examples are errors in article (Han et al., 2006; Nagata et al., 2006; Rozovskaya and Roth, 2011), preposition (Chodorow et al., 2007; Felice and Pulman, 2008; Rozovskaya and Roth, 2011; Tetreault et al., 2010), and tense (Nagata and Kawai, 2011; Tajiri et al., 2012), to name a few. Recently, Wu and Ng (2013) and Rozovskaya and Roth (2013) proposed methods for simultaneously correcting multiple types of errors using integer linear programming. Another major approach is to use a language model (LM) for predicting correct words or phrases for a given context. Some researchers (Brockett et al., 2006; Yoshimoto et al., 2013) use statistical machine translation (SMT) for the same purpose, which can be regarded as the mixture of a classifier and an LM. With these diverse techniques, correction performance has dramatically improved against a wide variety of target errors.

As noted above, however, one of the crucial limitations of these previous methods is that they

*Part of this work was performed while the author was a visiting researcher at LIMSI, Orsay (France).

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 754–764,
Baltimore, Maryland, USA, June 23-25 2014.

are not capable of providing feedback messages. They are not suitable for generating open-class text such as feedback messages by their nature. Some researchers (Kakegawa et al., 2000; McCoy et al., 1996) made an attempt to develop hand-crafted rules for correcting errors with feedback messages. However, this approach encounters the tremendous difficulty of covering a wide variety of errors using hand-crafted rules.

In view of this background, this paper presents a novel error correction framework called *error case frames* an example of which is shown in Fig. 2. They are case frames specially designed for describing and correcting errors in preposition attached to a verb; the reader may be able to see that it describes preposition errors such as **John often goes shopping to the market with his family.* and that the preposition *to* should be replaced with *at*. This paper proposes a method for automatically generating them by comparing learner and native corpora. Achieving a comparable correction performance, they have the following two advantages over the conventional approaches: (i) they are intuitively interpretable and manually modifiable to enrich them; (ii) they are capable of providing feedback messages.

The rest of this paper is structured as follows. Sect. 2 introduces the definition of error case frames. Sect. 3 discusses the method for generating error case frames. Sect. 4 describes how to correct preposition errors with feedback messages by error case frames. Sect. 5 describes experiments conducted to evaluate error case frames. Sect. 6 discusses the experimental results.

2 Error Case Frame

An error case frame consists of a verb, cases, and a feedback message as shown in Fig. 2[1]. The following explains error case frames in detail based on this example; occasionally consulting it may help understanding the following sections.

An error case frame always has a verb. In Fig. 2, the verb is *go*.

Cases are arguments the verb takes in an error case frame. A case consists of a case tag and case elements. A case tag and case elements describe, respectively, the role that the case plays in the error case frame and a set of words that are allowed

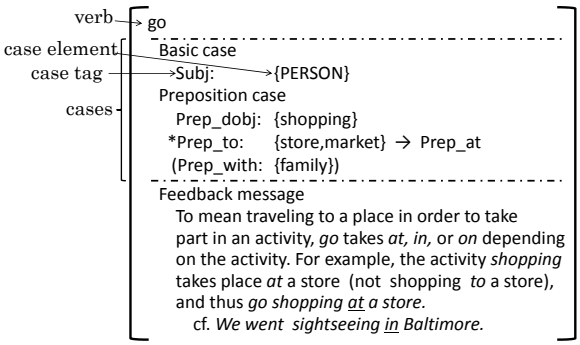


Figure 2: Example of an error case frame.

to appear as the argument. For instance, in Fig. 2, "Subj: {PERSON}" is a case where its case tag and element are "Subj:" and "{PERSON}," respectively, denoting that a person such as *John* plays a role of the subject of the verb. Note that tokens in all upper case such as "PERSON" refer to a group of words such as {john,he,$\cdots$} in this paper.

Cases are classified into two categories: basic and preposition cases. Basic cases are either a subject or a particle, whose case tags are "Subj:" and "Ptr:", respectively. The "Subj:" case is obligatory while the "Ptr:" is optional. Preposition cases correspond to the prepositions the verb takes as its arguments. Its case tag has the form of "Prep_x" where x ranges over the target prepositions. It should be emphasized that direct and indirect objects are included in the preposition cases for efficiency; their case tags are denoted as "Prep_dobj" and "Prep_iobj", respectively. Preposition cases are classified into those obligatory and optional. *Optional* here means that the verb can constitute a sentence with or without the preposition. Optional prepositions are written in parentheses as in "(Prep_with:{family})".

Preposition cases describe the information about an error. An error case frame is constrained to contain only one erroneous preposition case. It is marked with the symbol "*". So, the preposition case "*Prep_to:{store,market}" is erroneous in Fig. 2. The correct preposition is described after the symbol "→" as in "→ Prep_at".

Error case frames are furnished with feedback messages. Unlike verbs and cases, which are automatically filled based on corpus data, they are manually edited. A human annotator interprets error case frames and adds explanatory notes to them. This may seem time-consuming. How-

[1]Fig. 2 shows an example of error case frames for illustration purposes. They are formally expressed in a machine-readable format such as XML.

ever, the editing is far more efficient than manually creating correction rules with feedback messages from scratch because error case frames are highly abstracted as explained in Sect. 3. Above all, it is a significant advantage over the previous classifier-/LM-based methods considering that there exists no effective technique for augmenting these methods with feedback messages.

3 Generating Error Case Frames

The method proposed here exploits two sources of corpus data: native and learner corpora. Case frames (error case frames without the information about an error and a feedback message) can be automatically extracted from parsed sentences as Kawahara and Uchimoto (2008) show. The proposed method generates error case frames by comparing case frames generated from the learner corpus with those from the native corpus. The basic approach is to extract, as error case frames, case frames which appear in the learner corpus but not in the native corpus. However, this approach is so simple that it extracts undesirable false error case frames which do not actually correspond to preposition errors. To overcome the problem, the following procedures are applied:
(1) Filtering input sentences
(2) Extracting case frames
(3) Recognizing optional cases
(4) Grouping case frames
(5) Selecting candidate error case frames
(6) Determining correct prepositions
(7) Enriching error case frames
(8) Manually editing error case frames

(1) Filtering input sentences: This is a pre-process to filter out unsuitable input sentences for case frame generation. Accurate parsing is essential for accurate case frame generation. Parsing errors tend to occur in longer sentences. To reduce parsing errors, Kawahara and Uchimoto (2008) propose filtering out sentences which are longer than 20 words. We adopt this filtering in our method. We also filter out sentences containing commas, which often introduce complex structures. We apply the filtering pre-process only to the native corpus; the availability of learner corpora is still somewhat limited and therefore we use all the sentences available in the learner corpus for better coverage of preposition errors.

(2) Extracting case frames: This procedure can be viewed as a slot filling task where the slots are the verb and the cases in a case frame. To achieve this, the corpus data are first parsed by a parser. Then, for each verb, the predicate-argument structures are extracted from the parses as shown in Fig. 3. Here, only head words are extracted as arguments. They are reduced to their base form when extracted. Certain classes of words are replaced with their corresponding sense (e.g., *John* to *PERSON*); the mapping between words and their senses is shown in Appendix A. In the case of the learner corpus, mis-spelt words are automatically corrected using a spell-checker. Finally, a case frame is created by filling its slots with the extracted predicate-argument structures. Hereafter, case frames generated from the native and learner corpora will be referred to as the native and learner case frames, respectively.

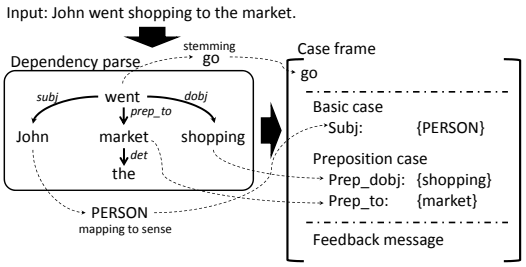


Figure 3: Example of case frame extraction.

(3) Recognizing optional cases: it is crucial for generating flexible error case frames to recognize optional preposition cases. Optional preposition cases are determined by the following heuristic rules: (a) Objects are always obligatory; (b) The number of obligatory preposition cases (except objects) is at most one; (c) Prepositions appearing left of the verb are optional; (d) Prepositions appearing right of the verb are optional except the one which is nearest to the verb. Rule (a) states that objects are always recognized as obligatory[2]. Rule (b) constrains an error case frame to have at most one obligatory preposition. Certain verbs sometimes have more than one obligatory preposition as in *range from A to B*. However, the large majority of verbs satisfy rule (b). Rule (c) states that prepositions appearing left of the verb

[2]A sentence can be constituted without objects as in *We sing.* Rule (a) always mistakenly recognizes such objects as obligatory. However, preposition errors never appear in sentences consisting of no object nor prepositions, and thus, the objects mistakenly recognized as obligatory never cause any problems in preposition error correction in practice.

in the input sentence are optional preposition cases as in *In the morning, he went shopping.* Rule (c) is based on the assumption that obligatory cases are tied to the verb more strongly than optional cases. In other words, obligatory cases cannot easily change their position. Conversely, optional cases have more freedom of their position, which enables them to appear left of a verb. Admittedly, obligatory prepositions can appear left of a verb as in *To school, he went* in certain circumstances such as in poetry. However, this usage is not so frequent in corpora normally used as training data such as newspaper articles. Rule (d), together with rule (b), states that if more than one preposition appears right of the verb, the one nearest to the verb is obligatory and the rest are optional. Rule (d) is based on the same reasoning as in rule (c).

Optional preposition cases are sometimes determined naturally by comparing two case frames. In this case, one of them must consist of only the object(s) as its preposition case(s) as in "[go Subj:{PERSON} Prep_dobj:{shopping}]." Then, the other case frame must consist of the same verb, the same basic cases, and the same object(s). The only difference between them is preposition cases (except the object(s)) (e.g., [go Subj:{PERSON} Prep_dobj:{shopping} Prep_at:{market}]). The case frame only with the object(s) proves the other to be valid without the preposition case(s). Thus, these preposition cases are recognized as optional (e.g., [go Subj:{PERSON} Prep_dobj:{shopping} (Prep_at:{market})]).

(4) Grouping case frames: Similar case frames in the native case frames are grouped into one, which will play an important role in **(7) Enriching error case frames**. Case frames comprising similar cases tend to denote similar usage of a verb. Considering this, case frames are merged into one if they consist of the same verb, the same basic cases, and the same case tags of the obligatory preposition cases. The grouping procedure is illustrated in Fig. 4. When preposition cases are obligatory in one case frame and optional in the other, the discrepancy is resolved by setting the preposition case to optional in the merged case frame. Note that this grouping procedure is not applied to the learner case frames so that erroneous usages in the learner case frames do not propagate to other (correct) learner case frames.

(5) Selecting candidate error case frames: Candidates for error case frames are selected from the learner case frames. If a learner case frame does not match, ignoring optional preposition cases, any native case frame, it is selected as a candidate for an error case frame on the assumption that case frames corresponding to erroneous usages do not appear in the native corpus.

Alternatively, an error-annotated learner corpus can be used to select error case frames; simply extracting case frames of which preposition is marked as an error gives error case frames. In this case[3], procedure (6) may be omitted and procedure (7) is directly applied after procedure (5).

(6) Determining correct prepositions: Now, correct prepositions for the candidate error case frames are explored. Each case tag of the preposition cases in a candidate is replaced, one at a time, with one of the other target prepositions. This replacement can be interpreted as error correction. Take as an example the following candidate error case frame: [go Subj:{PERSON} Prep_dobj:{shopping} Prep_to:{market}]. Replacing the case tag "Prep_to" with "Prep_at" corresponds to correct expressions such as *John often goes shopping at the market.* Note that replacing a direct object with one of the prepositions corresponds to correcting an omission error as in "Prep_dobj" with "Prep_to" in "[go Subj:{PERSON} Prep_dobj:{market}]". Similarly, replacing a preposition with an object corresponds to correcting an extra-preposition error (e.g., "Prep_to" with "Prep_dobj" in "[go Subj:{PERSON} Prep_to:{shopping}])".

To examine whether each correction is valid or not, the native case frames are again used; if the replaced case frame matches one of the native case frames, the correction is determined to be valid. Here, we define the match as the two case frames consisting of the same verb, the same basic cases, the same obligatory preposition cases, and the same preposition case to which the correction is applied (if it is an optional one). If the condition is satisfied, the information on the error and correction is added to the candidate error case frame. If a valid correction is found, the candidate is determined to be a valid error case frame. In total, their validity is double-checked, once in (5) and once in (6), by comparing them with the

[3] We do not make use of error-annotated learner corpora in this paper in order to reveal how well the proposed methods perform without such corpora. In practice, one can use error-annotated learner corpora together with raw learner corpora to achieve better performance.

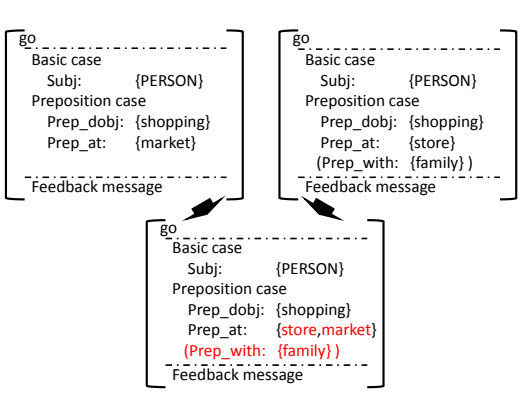


Figure 4: Example of grouping case frames.

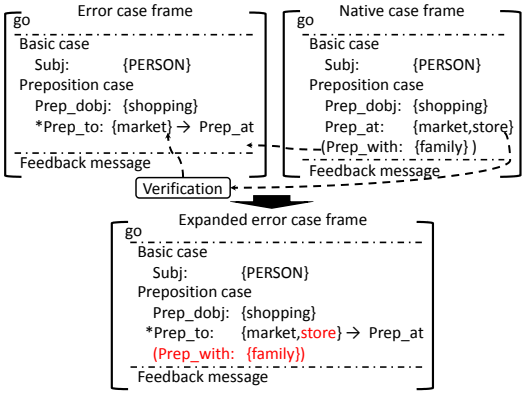


Figure 5: Enriching an error case frame.

native case frames.

(7) Enriching error case frames: The generated error cases are limited in error coverage because the procedures so far solely rely on preposition errors appearing in the learner corpus. In other words, it is impossible to generate error case frames corresponding to preposition errors which do not appear in the learner corpus. To overcome this limitation, the generated error case frames are enriched using the native case frames. For each error case frame, we already know the corresponding native (thus, correct) case frame, which is obtained in (6). The corresponding native case frame is normally much richer in preposition cases because of the optional cases and grouping given by procedures (3) and (4), as shown at the top of Fig. 5. These additional cases are useful to enrich error case frames.

For the preposition case which is determined to be erroneous, its correct preposition is found in the error case frame (e.g., "→ Prep_at" at the top-left of Fig. 5). Also, its correct preposition case is found in the corresponding native case frame (e.g., "Prep_at:{market,store}" at the top-right). Replacing the case element of the erroneous case by one of the case elements of the correct preposition case gives a new candidate for an error case frame (e.g., replacing *market* of "*Prep_to:{market}" by *store* gives "[go Subj:{PERSON} Prep_dobj:{shopping} *Prep_to:{store}]." It should be emphasized that this new error case frame is still a candidate at this point and the usage might be correct. To verify if it really describes an erroneous preposition use, the native case frames are searched for; if it matches one of them, that means that the use of the preposition actually appears in the native corpus. Therefore, it should be discarded. Only if a match is not found, is the case element added to the erroneous preposition case in the original error case frame. This process is illustrated in the box denoted as *Verification* in Fig. 5.

For the other preposition cases which are not erroneous, the enriching procedure is much simpler. They are simply added to the error case frame as shown in Fig. 5. One thing we should take care of is that there might be a discrepancy in obligatory/optional between the cases of the error case frame and the native case frame. This discrepancy is solved by setting the preposition case in the error case frame to optional. The resulting expanded error case frame after procedure (7) is shown at the bottom of Fig. 5 where the enriched cases are shown in red.

(8) Manually Editing Error Case Frames: The most important editing is the addition of feedback messages. A human annotator interprets the generated error case frames and adds explanatory notes to them. Although this basically requires manual editing, part of feedback messages can be automatically created to facilitate the procedure. For example, example sentences corresponding to an error case frame can be automatically added to it, whether correct or error examples, because the original sentences from which the (error) case frames extracted are available in the native and learner corpora. Besides, setting a variable to the feedback message allows it to be adaptable to correction results as shown in Fig. 6. In Fig. 6, $X_{\text{Prep\_to}}$ is a variable. It is replaced with one of the case elements of "Prep_to:" depending on correction results. Also, it will be beneficial to link similar error case frames each other, which allows the user to obtain additional information.

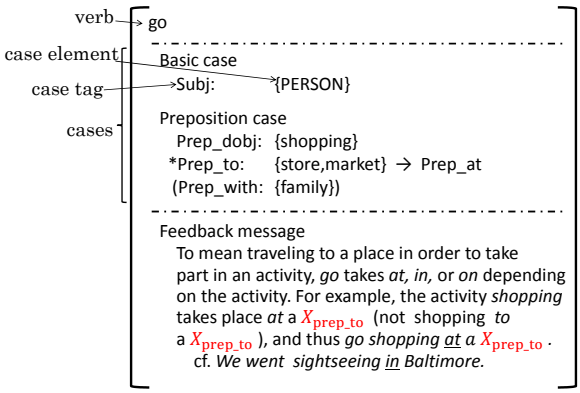


Figure 6: Error case frame with a variable.

For example, the example error case frame in Fig. 6 may be linked to similar case frames such as "[go Subj:{PERSON} Prep_dobj:{sightseeing} *Prep_to:{Baltimore} → Prep_in]." One can retrieve similar error case frames from the generated error case frames where the similarity between two error case frames are defined by the overlap in the verb, the basic cases, and the case tags of the preposition cases.

The generated error case frames may be further edited to enrich them. As we can see in Fig. 5, the generated error case frames are easy to interpret. This property enables us to manually edit them to enrich their preposition cases. For example, one might add a case element such as *supermarket* to the preposition case "Prep_to:{market,store}" in the example error case frame. Conversely, one might discard unnecessary case elements, cases, or even error case frames.

4 Correcting Preposition Errors

Preposition errors are corrected by applying the generated error case frames to the target text. Case frames are first extracted from the target text by the same procedures (2) and (3) in Sect. 3. Then, each extracted case frame is examined if it matches one of the error case frames. If a match is found, the preposition is detected as an error and the correct preposition is suggested with the feedback message according to the matched error case frame. The match between a case frame and an error case frame is defined in the exact same manner as in procedure (4) in Sect. 3. Sometimes, a case frame matches more than one error case frame suggesting different corrections. In this case, the most frequent correction among the candidates is chosen to correct the error, which was applied in the evaluation described in Sect. 5[4].

One of the advantages of error case frames is that they do not require an error-annotated corpus as explained in the previous section. This means that the target text itself can be used as part of a learner corpus for generating error case frames at the time of error correction. Applying procedures (2) to (7) to the target text generates additional error case frames[5]. Although feedback messages are not available in these additional error case frames, they are still useful for improving correction performance, especially in recall. Hereafter, this way of error case frame generation will be referred to as *active generation*.

A pre-experiment using a development data set revealed that there were some preposition errors for which error case frames were not generated even though the corresponding erroneous and correct preposition usages appeared in the learner and native corpora, respectively. They are preposition errors where the preposition is incorrectly used with an adverb as in **John went to there*. To be precise, they are either an adverb denoting a place (e.g., *there*) with a preposition concerning a place (*at*, *in*, *on*, and *to*) or a noun denoting time, frequency, and duration with a preposition concerning time, frequency, and duration (*at*, *for*, *in*, and *on*). In the native corpus, these adverbs or nouns are correctly used without a preposition and thus they are not recognized as a prepositional phrase by a parser. Therefore, corresponding native case frames are never found for these types of errors in procedure (6), and in turn error case frames are never generated for them.

Considering that they are limited in number because they are independent of verbs and basic cases, we decided to manually create error case frames describing these types of errors. In these error case frames, the verb and the basic cases are filled with ANY denoting any word. The preposition cases are manually filled based on the linguistic knowledge known as *absence of preposition* (Quirk et al., 1985). For example, an error case frame for the above error would be "[ANY Subj:{ANY} *Prep_to:{here,somewhere,there} → Prep_dobj]." Certain errors involve a phrase such as **John goes shopping in every morning*. To handle these cases,

[4]Ties are broken by random selection.

[5]Recall that procedure (1) is only applied to the native corpus.

these manually created error case frames are allowed to have phrases as their case elements (e.g., [ANY Subj:{ANY} *Prep_in:{every morning} → Prep_dobj]).

5 Evaluation

We evaluated the proposed method from two points of view: correction performance and usefulness of feedback messages. We measured correction performance by recall, precision, and F-measure. In the evaluation on usefulness of feedback messages, three human raters (a teacher of English at college and two who have a master degree in TESOL) separately examined whether each feedback message was useful for learning the correct usage of the preposition. We defined usefulness by the ratio of feedback messages evaluated as useful to the total number of feedback messages.

We used the following data sets in the evaluation. We selected the Konan-JIEM (KJ) learner corpus (Nagata et al., 2011) as the target texts. The KJ learner corpus is fully annotated with grammatical errors. In addition, it includes error correction results of several benchmark systems. This means that one can directly compare correction results of a new method with those of the benchmark systems, which reveals where the method is strong and weak compared to the benchmark systems. The KJ corpus consists of training and test sets. We used the training set to generate error case frames and evaluated correction performance on the test set. In addition to these data sets, we created a development set, which we had collected to develop the proposed method. We did not use it in the final evaluation. As a native corpus, we used the EDR corpus (Japan electronic dictionary research institute Ltd, 1993), the Reuters-21578 corpus[6], and the LOCNESS corpus[7]. We used the lexicalized dependency parser in the Stanford Statistical Natural Language Parser (ver.2.0.3) (de Marneffe et al., 2006) to obtain parses for the data sets. Table 1 shows the statistics on the data sets.

Using these data sets, we implemented three versions of the proposed method. The first one was based on error case frames generated from the training set of the KJ corpus. The second one was the first one with active generation. To implement the third one, we manually edited the error case frames of the first version to remove unnecessary error case frames and case elements (but no addition) and to add feedback messages to them. After this, active generation was applied to augment the edited error case frames. In implementing the proposed methods, we selected as target prepositions the ten most frequent prepositions, the same as in previous work (Rozovskaya and Roth, 2011): *about*, *at*, *by*, *for*, *from*, *in*, *of*, *on*, *to*, *with*.

Name	# of tokens	# of errors
KJ training	22,701	327
KJ test	8,065	131
Dev. set	47,217	774
EDR	1,745,863	—
Reuters	28,431,228	—
LOCNESS	294,325	—

Table 1: Statistics on the data sets for evaluation.

For comparison, we selected two conventional methods. One was the best-performing system among the benchmark systems, which is the classifier-based method (Sakaguchi et al., 2012) which had participated in the HOO 2012 shared task (Dale et al., 2012). The other was the SMT-based method (Yoshimoto et al., 2013) which was the best-performing system in preposition error correction in the CoNLL 2013 shared task (Ng et al., 2013). In addition, we evaluated performance of hybrid methods combining the correction results of the third version of the proposed method with those of the classifier-/SMT-based method; we simply took the union of the two.

Table 2 shows the evaluation results. The simple error case frame-based method achieves an F-measure of 0.189. It improves recall when combined with active generation, which shows the effectiveness of active generation for augmenting error case frames. It further improves precision without decreasing recall by manual editing; note that manual editing was only applied to the error case frames generated from the training data but not to those generated by active generation. The performance is comparable to both classifier-/SMT-based methods. The hybrid methods achieve the best performances in F-measure.

In the usefulness evaluation, the third version of the proposed method was able to provide 20 feedback messages for the target texts. The three human raters evaluated 80%, 80%, and 85% of the

[6]Reuters-21578, Distribution 1.0, `http://www.research.att.com/~lewis`

[7]`http://www.uclouvain.be/en-cecl.html`

Method	R	P	F
ECF	0.107	**0.823**	0.189
ECF with AG	0.130	0.680	0.218
ME-ECF with AG	0.130	0.708	0.219
Classifier-based	0.167	0.310	0.217
SMT-based	0.115	0.385	0.176
Classifier hybrid	**0.235**	0.369	**0.287**
SMT hybrid	0.191	0.446	0.267

ECF: Error Case Frame, ME-ECF: Manually Edited Error Case Frame, AG: Active Generation

Table 2: Correction performance in recall (R), precision (P), and F-measure (F).

20 feedback messages as useful (82% on average). The agreement among the raters was $\kappa = 0.67$ in Fleiss's κ.

6 Discussion

As the experimental results show, the proposed method achieves a comparable correction performance with the classifier-/SMT-based methods. A closer look at the correction results reveals the differences in correction tendencies between these methods, which explains well why the hybrid methods achieve better performance.

One of the tendencies is that the proposed method performs better on preposition errors where relatively wider contexts are required to correct them. Error case frames naturally exploit wider contexts based on the cases which are extracted by parsing. In contrast, classifier-/SMT-based methods rely on narrower contexts such as a few words surrounding the preposition in question. Take as an example the following sentence which appeared in the test set: **In the univerysity, I studied English in the morning*[8]. To confirm that the preposition *In* is erroneous requires the verb *studied* and the object *English*. The proposed method successfully corrected this error by the error case frame "[study Subj:{PERSON} Prep_dobj:{english,math,$\cdots$} *Prep_in:{university} $\rightarrow$ at]" in the evaluation. This would be difficult for methods relying on only a few words surrounding the preposition *In*.

It is also difficult for classifier-/SMT-based methods to correct missing preposition errors. Classifier-based methods need to be informed of the position of the preposition to predict a correct preposition. Because the position of a missing preposition is implicit, classifier-based methods would have to make a prediction at every single position between words, which would be inefficient. Because of this, the classifier-based method used in the evaluation (and often other classifier-based methods) excludes missing preposition errors from its target. SMT-based methods do not perform well either on missing preposition errors because of the fact that they implicitly, but not directly, handle missing preposition errors. In contrast, error case frames directly model missing prepositions by treating objects as one of the preposition cases (i.e., *Prep_dobj*).

Grammatical errors other than preposition errors influence both the proposed and classifier-/SMT-based methods, but differently. Grammatical errors appearing around the preposition in question seem to influence the previous methods more significantly than the proposed method because they rely on words surrounding the preposition. On the other hand, structural errors such as errors in voice tend to degrade performance of the proposed method. For instance, if an error in voice occurs as in **I excited _ this*, correctly, *I was excited by this*, error case frames are not properly applied.

The precisions of the proposed methods are high compared to those of the previous methods. To be precise, the number of false positives is only seven in the third version of the proposed method. Out of seven, four false positives are due to problems with the used error case frames themselves. Two are the influence of other grammatical errors (e.g., **I like to look beautiful view.* was corrected as *look at beautiful view* by the proposed method but as *see beautiful view* in the error annotation).

Unlike false positives, it is difficult to precisely point out causes for false negatives, which often involve several factors. One cause which is theoretically clear is errors in preposition attached to a noun phrase (NP), which amounts to 11 % of all false negatives. Since error case frames describe errors in preposition attached to a verb, they do not target these types of errors. Extending error case frames to general frames might overcome this limitation, which will require further investigation. Similarly, error case frames are not generated for preposition errors where prepositions are incorrectly used with words other than a noun as

[8]The word *univerysity* is a mis-spelt word of *university*. Note that mis-spelt words are automatically corrected by a spell-checker when case frames are extracted.

in **make me to happy* (5 % of all). Although error case frames can describe these types of errors, case frames are not extracted for their corresponding correct usages from the native corpus. This is because the word in question (e.g., *happy*) correctly appears without the erroneous preposition in the native corpus, and thus it is not recognized as a preposition case. This means that a corresponding correct case frame is never found for any error of these types in the generation procedure (6). Accordingly, error case frames are never generated for these types of errors. The most influential cause of false negatives, which is also a major cause of false negatives in the previous methods, is other grammatical errors (at least 22 % of all). One of such errors is errors in voice as already explained (4%). Another is the omission of the object of a verb (4%). In these cases, even if an appropriate error case frame exists, it is not applied because of the grammatical error.

In addition to correction performance, error case frames are effective in providing feedback messages; Fig. 1 (on the first page) shows excerpts of the feedback messages provided in the evaluation. The evaluation shows that 82% of the provided feedback messages were actually rated as useful for language learning on average (the rest were mostly evaluated as not-useful due to false positive corrections). With the feedback messages of error case frames, we now have the following three choices as the way of error correction: (a) just indicating the correct preposition (as in previous methods); (b) indicating the correction preposition with a feedback message; (c) displaying only a feedback message. In (a), the learner might just copy the correct preposition to correct his or her writing, which would result in little or no learning effect. This suggests that the ultimate goal of grammatical error correction for language learning assistance is not to correct all errors in the given text but to maximize learning effect for the learner. (b) might give a similar result because the learner can copy the correct preposition without reading the feedback message. In (c), the learner has to actually read and understand the feedback message to select the correct preposition. Taking these into consideration, (c) will likely give the learner better learning effect than the other two. Therefore, we propose applying the feedback (c) to language learning assistance. To the best of our knowledge, it is only the error case frame-based method that is capable of this manner of error correction.

7 Conclusions

This paper presented a novel framework called error case frames for correcting preposition errors with feedback messages. The evaluation showed that (i) automatically generated error case frames achieve a performance comparable to conventional methods; (ii) they are intuitively interpretable and manually modifiable to improve them; (iii) feedback messages provided by error case frames are effective in language learning assistance. Considering these advantages and the fact that it has been difficult to provide feedback messages by automatically generated rules, error case frames will likely be one of the major approaches for preposition error correction.

Appendix A. Sense mapping

The following list shows the mapping between words and senses developed based on the WordNet (Miller, 1995) and GSK dictionary of places and facilities (2nd Ed.)[9]. Each line consists of a token for a sense, its definition, examples of its member. DRINK (drink): tea, coffee
FOOD (food): cake, sandwich
MONTH (names of months): January, February
MINST (musical instruments): guitar, piano
PERSON (persons): John, he
PLACE (place names): Canada, Paris
SPORT (sports): football, tennis
SPORTING (sporting activities): swimming
WEEK (the days of the week): Monday
VEHICLE (vehicles): train, bus

Acknowledgments

We would like to thank Daisuke Kawahara for his advice on case frame generation. We also would like to thank Keisuke Sakaguchi and Mamoru Komachi for providing the authors with their system outputs. Finally, we would acknowledge the help from the members of the ILES group at LIMSI, Orsay (France) where the first author performed part of this work. This work was partly supported by Kaken Grant-in-Aid for Young Scientists (B) (26750091).

[9]GSK dictionary of places and facilities second edition: http://www.gsk.or.jp/en/catalog/gsk2012-c/

References

Chris Brockett, William B. Dolan, and Michael Gamon. 2006. Correcting ESL errors using phrasal SMT techniques. In *Proc. of 21th International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 249–256, Sydney, Australia, July.

Martin Chodorow, Joel R. Tetreault, and Na-Rae Han. 2007. Detection of grammatical errors involving prepositions. In *Proc. of 4th ACL-SIGSEM Workshop on Prepositions*, pages 25–30.

Robert Dale, Ilya Anisimoff, and George Narroway. 2012. HOO 2012: A report on the preposition and determiner error correction shared task. In *Proc. 7th Workshop on Building Educational Applications Using NLP*, pages 54–62.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proc. of 5th International Conference on Language Resources and Evaluation*, pages 449–445.

Rachele De Felice and Stephen G. Pulman. 2008. A classifier-based approach to preposition and determiner error correction in L2 English. In *Proc. of 22nd International Conference on Computational Linguistics*, pages 169–176.

Dana Ferris and Barrie Roberts. 2001. Error feedback in L2 writing classes: How explicit does it need to be? *Journal of Second Language Writing*, 10(3):161–184.

Na-Rae Han, Martin Chodorow, and Claudia Leacock. 2006. Detecting errors in English article usage by non-native speakers. *Natural Language Engineering*, 12(2):115–129.

Japan electronic dictionary research institute Ltd. 1993. *EDR electronic dictionary specifications guide*. Japan electronic dictionary research institute ltd.

Jun'ichi Kakegawa, Hisayuki Kanda, Eitaro Fujioka, Makoto Itami, and Kohji Itoh. 2000. Diagnostic processing of Japanese for computer-assisted second language learning. In *Proc. of 38th Annual Meeting of the Association for Computational Linguistics*, pages 537–546.

Daisuke Kawahara and Kiyotaka Uchimoto. 2008. A method for automatically constructing case frames for English. In *Proc. of 6th International Conference on Language Resources and Evaluation*.

Kathleen F. McCoy, Christopher A. Pennington, and Linda Z. Suri. 1996. English error correction: A syntactic user model based on principled "mal-rule" scoring. In *Proc. of 5th International Conference on User Modeling*, pages 69–66.

George A. Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.

Ryo Nagata and Atsuo Kawai. 2011. Exploiting learners' tendencies for detecting English determiner errors. In *Lecture Notes in Computer Science*, volume 6882/2011, pages 144–153.

Ryo Nagata, Atsuo Kawai, Koichiro Morihiro, and Naoki Isu. 2006. A feedback-augmented method for detecting errors in the writing of learners of English. In *Proc. of 44th Annual Meeting of the Association for Computational Linguistics*, pages 241–248.

Ryo Nagata, Edward Whittaker, and Vera Sheinman. 2011. Creating a manually error-tagged and shallow-parsed learner corpus. In *Proc. of 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1210–1219.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction. In *Proc. 17th Conference on Computational Natural Language Learning: Shared Task*, pages 1–12.

Randolph Quirk, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik. 1985. *A Comprehensive Grammar of the English Language*. Longman, New York.

Thomas Robb, Steven Ross, and Ian Shortreed. 1986. Salience of feedback on error and its effect on EFL writing quality. *TESOL QUARTERY*, 20(1):83–93.

Alla Rozovskaya and Dan Roth. 2011. Algorithm selection and model adaptation for ESL correction tasks. In *Proc. of 49th Annual Meeting of the Association for Computational Linguistics*, pages 924–933.

Alla Rozovskaya and Dan Roth. 2013. Joint learning and inference for grammatical error correction. In *Proc. of Conference on Empirical Methods in Natural Language Processing*, pages 791–802.

Keisuke Sakaguchi, Yuta Hayashibe, Shuhei Kondo, Lis Kanashiro, Tomoya Mizumoto, Mamoru Komachi, and Yuji Matsumoto. 2012. NAIST at the HOO 2012 shared task. In *Proc. of 7th Workshop on the Innovative Use of NLP for Building Educational Applications*, pages 281–288.

Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. 2012. Tense and aspect error correction for ESL learners using global context. In *Proc. of 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 198–202.

Joel Tetreault, Jennifer Foster, and Martin Chodorow. 2010. Using parse features for preposition selection and error detection. In *Proc. of 48nd Annual Meeting of the Association for Computational Linguistics Short Papers*, pages 353–358.

Yuanbin Wu and Hwee Tou Ng. 2013. Grammatical error correction using integer linear programming. In *Proc. of 51st Annual Meeting of the Association for Computational Linguistics*, pages 1456–1465.

Ippei Yoshimoto, Tomoya Kose, Kensuke Mitsuzawa, Keisuke Sakaguchi, Tomoya Mizumoto, Yuta Hayashibe, Mamoru Komachi, and Yuji Matsumoto. 2013. NAIST at 2013 CoNLL grammatical error correction shared task. In *Proc. of 17th Conference on Computational Natural Language Learning: Shared Task*, pages 26–33.

Kneser-Ney Smoothing on Expected Counts

Hui Zhang
Department of Computer Science
University of Southern California
hzhang@isi.edu

David Chiang
Information Sciences Institute
University of Southern California
chiang@isi.edu

Abstract

Widely used in speech and language processing, Kneser-Ney (KN) smoothing has consistently been shown to be one of the best-performing smoothing methods. However, KN smoothing assumes integer counts, limiting its potential uses—for example, inside Expectation-Maximization. In this paper, we propose a generalization of KN smoothing that operates on fractional counts, or, more precisely, on *distributions* over counts. We rederive all the steps of KN smoothing to operate on count distributions instead of integral counts, and apply it to two tasks where KN smoothing was not applicable before: one in language model adaptation, and the other in word alignment. In both cases, our method improves performance significantly.

1 Introduction

In speech and language processing, smoothing is essential to reduce overfitting, and Kneser-Ney (KN) smoothing (Kneser and Ney, 1995; Chen and Goodman, 1999) has consistently proven to be among the best-performing and most widely used methods. However, KN smoothing assumes integer counts, whereas in many NLP tasks, training instances appear with possibly fractional weights. Such cases have been noted for language modeling (Goodman, 2001; Goodman, 2004), domain adaptation (Tam and Schultz, 2008), grapheme-to-phoneme conversion (Bisani and Ney, 2008), and phrase-based translation (Andrés-Ferrer, 2010; Wuebker et al., 2012).

For example, in Expectation-Maximization (Dempster et al., 1977), the Expectation (E) step computes the posterior distribution over possible completions of the data, and the Maximization (M) step reestimates the model parameters as if that distribution had actually been observed. In most cases, the M step is identical to estimating the model from complete data, except that counts of observations from the E step are fractional. It is common to apply add-one smoothing to the M step, but we cannot apply KN smoothing.

Another example is instance weighting. If we assign a weight to each training instance to indicate how important it is (say, its relevance to a particular domain), and the counts are not integral, then we again cannot train the model using KN smoothing.

In this paper, we propose a generalization of KN smoothing (called *expected KN* smoothing) that operates on fractional counts, or, more precisely, on *distributions* over counts. We rederive all the steps of KN smoothing to operate on count distributions instead of integral counts. We demonstrate how to apply expected KN to two tasks where KN smoothing was not applicable before. One is language model domain adaptation, and the other is word alignment using the IBM models (Brown et al., 1993). In both tasks, expected KN smoothing improves performance significantly.

2 Smoothing on integral counts

Before presenting our method, we review KN smoothing on integer counts as applied to language models, although, as we will demonstrate in Section 7, KN smoothing is applicable to other tasks as well.

2.1 Maximum likelihood estimation

Let $\mathbf{u}w$ stand for an n-gram, where $\mathbf{u}$ stands for the $(n-1)$ context words and w, the predicted word. Let $c(\mathbf{u}w)$ be the number of occurrences of $\mathbf{u}w$. We use a bullet ($\bullet$) to indicate summation over words, that is, $c(\mathbf{u}\bullet) = \sum_w c(\mathbf{u}w)$. Under maximum-likelihood estimation (MLE), we max-

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 765–774,
Baltimore, Maryland, USA, June 23-25 2014.

imize

$$L = \sum_{\mathbf{u}w} c(\mathbf{u}w) \log p(w \mid \mathbf{u}),$$

obtaining the solution

$$p_{\text{mle}}(w \mid \mathbf{u}) = \frac{c(\mathbf{u}w)}{c(\mathbf{u}\bullet)}. \tag{1}$$

2.2 Absolute discounting

Absolute discounting (Ney et al., 1994) – on which KN smoothing is based – tries to generalize better to unseen data by subtracting a *discount* from each seen n-gram's count and distributing the subtracted discounts to unseen n-grams. For now, we assume that the discount is a constant D, so that the smoothed counts are

$$\tilde{c}(\mathbf{u}w) = \begin{cases} c(\mathbf{u}w) - D & \text{if } c(\mathbf{u}w) > 0 \\ n_{1+}(\mathbf{u}\bullet) D q_{\mathbf{u}}(w) & \text{otherwise} \end{cases}$$

where $n_{1+}(\mathbf{u}\bullet) = |\{w \mid c(\mathbf{u}w) > 0\}|$ is the number of word types observed after context $\mathbf{u}$, and $q_{\mathbf{u}}(w)$ specifies how to distribute the subtracted discounts among unseen n-gram types. Maximizing the likelihood of the smoothed counts $\tilde{c}$, we get

$$p(w \mid \mathbf{u}) = \begin{cases} \dfrac{c(\mathbf{u}w) - D}{c(\mathbf{u}\bullet)} & \text{if } c(\mathbf{u}w) > 0 \\ \dfrac{n_{1+}(\mathbf{u}\bullet) D q_{\mathbf{u}}(w)}{c(\mathbf{u}\bullet)} & \text{otherwise.} \end{cases} \tag{2}$$

How to choose D and $q_{\mathbf{u}}(w)$ are described in the next two sections.

2.3 Estimating D by leaving-one-out

The discount D can be chosen by various means; in absolute discounting, it is chosen by the method of *leaving one out*. Given N training instances, we form the probability of each instance under the MLE using the other $(N-1)$ instances as training data; then we maximize the log-likelihood of all those instances. The probability of an n-gram token $\mathbf{u}w$ using the other tokens as training data is

$$p_{\text{loo}}(w \mid \mathbf{u}) = \begin{cases} \dfrac{c(\mathbf{u}w) - 1 - D}{c(\mathbf{u}\bullet) - 1} & c(\mathbf{u}w) > 1 \\ \dfrac{(n_{1+}(\mathbf{u}\bullet) - 1) D q_{\mathbf{u}}(w)}{c(\mathbf{u}\bullet) - 1} & c(\mathbf{u}w) = 1. \end{cases}$$

We want to find the D that maximizes the leaving-one-out log-likelihood

$$\begin{aligned} L_{\text{loo}} &= \sum_{\mathbf{u}w} c(\mathbf{u}w) \log p_{\text{loo}}(w \mid \mathbf{u}) \\ &= \sum_{\mathbf{u}w \mid c(\mathbf{u}w) > 1} c(\mathbf{u}w) \log \frac{c(\mathbf{u}w) - 1 - D}{c(\mathbf{u}\bullet) - 1} \\ &\quad + \sum_{\mathbf{u}w \mid c(\mathbf{u}w) = 1} \log \frac{(n_{1+}(\mathbf{u}\bullet) - 1) D q_{\mathbf{u}}(w)}{c(\mathbf{u}\bullet) - 1} \\ &= \sum_{r>1} r n_r \log(r - 1 - D) + n_1 \log D + C, \end{aligned} \tag{3}$$

where $n_r = |\{\mathbf{u}w \mid c(\mathbf{u}w) = r\}|$ is the number of n-gram types appearing r times, and C is a constant not depending on D. Setting the partial derivative with respect to D to zero, we have

$$\begin{aligned} \frac{\partial L_{\text{loo}}}{\partial D} &= -\sum_{r>1} \frac{r n_r}{r - 1 - D} + \frac{n_1}{D} \\ \frac{n_1}{D} &= \sum_{r>1} \frac{r n_r}{r - 1 - D} \geq \frac{2 n_2}{1 - D}. \end{aligned}$$

Solving for D, we have

$$D \leq \frac{n_1}{n_1 + 2 n_2}. \tag{4}$$

Theoretically, we can use iterative methods to optimize D. But in practice, setting D to this upper bound is effective and simple (Ney et al., 1994; Chen and Goodman, 1999).

2.4 Estimating the lower-order distribution

Finally, $q_{\mathbf{u}}(w)$ is defined to be proportional to an $(n-1)$-gram model $p'(w \mid \mathbf{u}')$, where $\mathbf{u}'$ is the $(n-2)$-gram suffix of $\mathbf{u}$. That is,

$$q_{\mathbf{u}}(w) = \gamma(\mathbf{u}) p'(w \mid \mathbf{u}'),$$

where $\gamma(\mathbf{u})$ is an auxiliary function chosen to make the distribution $p(w \mid \mathbf{u})$ in (2) sum to one.

Absolute discounting chooses $p'(w \mid \mathbf{u}')$ to be the maximum-likelihood unigram distribution; under KN smoothing (Kneser and Ney, 1995), it is chosen to make p in (2) satisfy the following constraint for all $(n-1)$-grams $\mathbf{u}'w$:

$$p_{\text{mle}}(\mathbf{u}'w) = \sum_{v} p(w \mid v\mathbf{u}') p_{\text{mle}}(v\mathbf{u}'). \tag{5}$$

Substituting in the definition of p_{mle} from (1) and p from (2) and canceling terms, we get

$$\begin{aligned} c(\mathbf{u}'w) &= \sum_{v \mid c(v\mathbf{u}'w) > 0} (c(v\mathbf{u}'w) - D) \\ &\quad + \sum_{v \mid c(v\mathbf{u}'w) = 0} n_{1+}(v\mathbf{u}'\bullet) D \gamma(v\mathbf{u}') p'(w \mid \mathbf{u}'). \end{aligned}$$

Solving for $p'(w \mid \mathbf{u}')$, we have

$$p'(w \mid \mathbf{u}') = \frac{\sum_{v|c(v\mathbf{u}'w)>0} 1}{\sum_{v|c(v\mathbf{u}'w)=0} n_{1+}(v\mathbf{u}'\bullet)\gamma(v\mathbf{u}')}.$$

Kneser and Ney assume the denominator is constant in w and renormalize to get an approximation

$$p'(w \mid \mathbf{u}') \approx \frac{n_{1+}(\bullet\mathbf{u}'w)}{n_{1+}(\bullet\mathbf{u}'\bullet)}, \tag{6}$$

where

$$n_{1+}(\bullet\mathbf{u}'w) = |\{v \mid c(v\mathbf{u}'w) > 0\}|$$
$$n_{1+}(\bullet\mathbf{u}'\bullet) = \sum_w n_{1+}(\bullet\mathbf{u}'w).$$

3 Count distributions

The computation of D and p' above made use of n_r and n_{r+}, which presupposes integer counts. But in many applications, the counts are not integral, but fractional. How do we apply KN smoothing in such cases? In this section, we introduce *count distributions* as a way of circumventing this problem.

3.1 Definition

In the E step of EM, we compute a probability distribution (according to the current model) over all possible completions of the observed data, and the expected counts of all types, which may be fractional. However, note that in each completion of the data, the counts are integral. Although it does not make sense to compute n_r or n_{r+} on fractional counts, it does make sense to compute them on possible completions.

In other situations where fractional counts arise, we can still think of the counts as expectations under some distribution over possible "realizations" of the data. For example, if we assign a weight between zero and one to every instance in a corpus, we can interpret each instance's weight as the probability of that instance occurring or not, yielding a distribution over possible subsets of the data.

Let X be a random variable ranging over possible realizations of the data, and let $c_X(\mathbf{u}w)$ be the count of $\mathbf{u}w$ in realization X. The expectation $E[c_X(\mathbf{u}w)]$ is the familiar fractional expected count of $\mathbf{u}w$, but we can also compute the probabilities $p(c_X(\mathbf{u}w) = r)$ for any r. From now on, for brevity, we drop the subscript X and understand $c(\mathbf{u}w)$ to be a random variable depending on X. The $n_r(\mathbf{u}\bullet)$ and $n_{r+}(\mathbf{u}\bullet)$ and related quantities also become random variables depending on X.

For example, suppose that our data consists of the following bigrams, with their weights:

(a)	fat cat	0.3
(b)	fat cat	0.8
(c)	big dog	0.9

We can interpret this as a distribution over eight subsets (not all distinct), with probabilities:

$\emptyset$	$0.7 \cdot 0.2 \cdot 0.1 = 0.014$
{a}	$0.3 \cdot 0.2 \cdot 0.1 = 0.006$
{b}	$0.7 \cdot 0.8 \cdot 0.1 = 0.056$
{a, b}	$0.3 \cdot 0.8 \cdot 0.1 = 0.024$
{c}	$0.7 \cdot 0.2 \cdot 0.9 = 0.126$
{a, c}	$0.3 \cdot 0.2 \cdot 0.9 = 0.054$
{b, c}	$0.7 \cdot 0.8 \cdot 0.9 = 0.504$
{a, b, c}	$0.3 \cdot 0.8 \cdot 0.9 = 0.216$

Then the count distributions and the $E[n_r]$ are:

	$r = 1$	$r = 2$	$r > 0$
$p(c(\text{fat cat}) = r)$	0.62	0.24	0.86
$p(c(\text{big dog}) = r)$	0.9	0	0.9
$E[n_r]$	1.52	0.24	

3.2 Efficient computation

How to compute these probabilities and expectations depends in general on the structure of the model. If we assume that all occurrences of $\mathbf{u}w$ are independent (although in fact they are not always), the computation is very easy. If there are k occurrences of $\mathbf{u}w$, each occurring with probability p_i, the count $c(\mathbf{u}w)$ is distributed according to the *Poisson-binomial distribution* (Hong, 2013). The expected count $E[c(\mathbf{u}w)]$ is just $\sum_i p_i$, and the distribution of $c(\mathbf{u}w)$ can be computed as follows:

$$p(c(\mathbf{u}w) = r) = s(k, r)$$

where $s(k, r)$ is defined by the recurrence

$$s(k, r) = \begin{cases} s(k-1, r)(1 - p_k) & \\ \quad + s(k-1, r-1)p_k & \text{if } 0 \le r \le k \\ 1 & \text{if } k = r = 0 \\ 0 & \text{otherwise.} \end{cases}$$

We can also compute

$$p(c(\mathbf{u}w) \ge r) = \max\left\{s(m, r), 1 - \sum_{r'<r} s(m, r')\right\},$$

the floor operation being needed to protect against rounding errors, and we can compute

$$E[n_r(\mathbf{u}\bullet)] = \sum_w p(c(\mathbf{u}w) = r)$$
$$E[n_{r+}(\mathbf{u}\bullet)] = \sum_w p(c(\mathbf{u}w) \ge r).$$

Since, as we shall see, we only need to compute these quantities up to a small value of r (2 or 4), this takes time linear in k.

4 Smoothing on count distributions

We are now ready to describe how to apply KN smoothing to count distributions. Below, we recapitulate the derivation of KN smoothing presented in Section 2, using the expected log-likelihood in place of the log-likelihood and applying KN smoothing to each possible realization of the data.

4.1 Maximum likelihood estimation

The MLE objective function is the expected log-likelihood,

$$\begin{aligned} E[L] &= E\left[\sum_{\mathbf{u}w} c(\mathbf{u}w) \log p(w \mid \mathbf{u})\right] \\ &= \sum_{\mathbf{u}w} E[c(\mathbf{u}w)] \log p(w \mid \mathbf{u}) \end{aligned}$$

whose maximum is

$$p_{\text{mle}}(w \mid \mathbf{u}) = \frac{E[c(\mathbf{u}w)]}{E[c(\mathbf{u}\bullet)]}. \tag{7}$$

4.2 Absolute discounting

If we apply absolute discounting to every realization of the data, the expected smoothed counts are

$$\begin{aligned} E[\tilde{c}(\mathbf{u}w)] &= \sum_{r>0} p(c(\mathbf{u}w) = r)(r - D) \\ &\quad + p(c(\mathbf{u}w) = 0) E[n_{1+}(\mathbf{u}\bullet)] D q_{\mathbf{u}}(w) \\ &= E[c(\mathbf{u}w)] - p(c(\mathbf{u}w) > 0) D \\ &\quad + p(c(\mathbf{u}w) = 0) E[n_{1+}(\mathbf{u}\bullet)] D q_{\mathbf{u}}(w) \end{aligned} \tag{8}$$

where, to be precise, the expectation $E[n_{1+}(\mathbf{u}\bullet)]$ should be conditioned on $c(\mathbf{u}w) = 0$; in practice, it seems safe to ignore this. The MLE is then

$$p(w \mid \mathbf{u}) = \frac{E[\tilde{c}(\mathbf{u}w)]}{E[\tilde{c}(\mathbf{u}\bullet)]}. \tag{9}$$

4.3 Estimating D by leaving-one-out

It would not be clear how to perform leaving-one-out estimation on fractional counts, but here we have a distribution over realizations of the data, each with integral counts, and we can perform leaving-one-out estimation on each of these. In other words, our goal is to find the D that maximizes the *expected* leaving-one-out log-likelihood, which is just the expected value of (3):

$$\begin{aligned} E[L_{\text{loo}}] &= E\Big[n_1 \log D + \sum_{r>1} r n_r \log(r - 1 - D) + C\Big] \\ &= E[n_1] \log D \\ &\quad + \sum_{r>1} r E[n_r] \log(r - 1 - D) + C, \end{aligned}$$

where C is a constant not depending on D. We have made the assumption that the n_r are independent.

By exactly the same reasoning as before, we obtain an upper bound for D:

$$D \leq \frac{E[n_1]}{E[n_1] + 2E[n_2]}. \tag{10}$$

In our example above, $D = \frac{1.52}{1.52+2\cdot 0.24} = 0.76$.

4.4 Estimating the lower-order distribution

We again require p' to satisfy the marginal constraint (5). Substituting in (7) and solving for p' as in Section 2.4, we obtain the solution

$$p'(w \mid \mathbf{u}') = \frac{E[n_{1+}(\bullet \mathbf{u}' w)]}{E[n_{1+}(\bullet \mathbf{u}' \bullet)]}. \tag{11}$$

For the example above, the estimates for the unigram model $p'(w)$ are

$$\begin{aligned} p'(\text{cat}) &= \tfrac{0.86}{0.86+0.9} \approx 0.489 \\ p'(\text{dog}) &= \tfrac{0.9}{0.86+0.9} \approx 0.511. \end{aligned}$$

4.5 Extensions

Chen and Goodman (1999) introduce three extensions to Kneser-Ney smoothing which are now standard. For our experiments, we used all three, for both integral counts and count distributions.

4.5.1 Interpolation

In interpolated KN smoothing, the subtracted discounts are redistributed not only among unseen events but also seen events. That is,

$$\tilde{c}(\mathbf{u}w) = \max\{0, c(\mathbf{u}w) - D\} + n_{1+}(\mathbf{u}\bullet) D p'(w \mid \mathbf{u}').$$

In this case, $\gamma(\mathbf{u})$ is always equal to one, so that $q_{\mathbf{u}}(w) = p'(w \mid \mathbf{u}')$. (Also note that (6) becomes an exact solution to the marginal constraint.) Theoretically, this requires us to derive a new estimate for D. However, as this is not trivial, nearly all implementations simply use the original estimate (4).

On count distributions, the smoothed counts become

$$\begin{aligned} E[\tilde{c}(\mathbf{u}w)] &= E[c(\mathbf{u}w)] - p(c(\mathbf{u}w) > 0) D \\ &\quad + E[n_{1+}(\mathbf{u}\bullet)] D p'(w \mid \mathbf{u}'). \end{aligned} \tag{12}$$

In our example, the smoothed counts are:

$\mathbf{u}w$	$E[\tilde{c}]$
fat cat	$1.1 - 0.86 \cdot 0.76 + 0.86 \cdot 0.76 \cdot 0.489 \approx 0.766$
fat dog	$0 \quad - 0 \cdot 0.76 \quad + 0.86 \cdot 0.76 \cdot 0.511 \approx 0.334$
big cat	$0 \quad - 0 \cdot 0.76 \quad + 0.9 \cdot 0.76 \cdot 0.489 \approx 0.334$
big dog	$0.9 - 0.9 \cdot 0.76 \quad + 0.9 \cdot 0.76 \cdot 0.511 \approx 0.566$

which give the smoothed probability estimates:

$$p(\text{cat} \mid \text{fat}) = \tfrac{0.766}{0.766+0.334} = 0.696$$
$$p(\text{dog} \mid \text{fat}) = \tfrac{0.334}{0.766+0.334} = 0.304$$
$$p(\text{dog} \mid \text{big}) = \tfrac{0.334}{0.334+0.556} = 0.371$$
$$p(\text{cat} \mid \text{big}) = \tfrac{0.556}{0.334+0.556} = 0.629.$$

4.5.2 Modified discounts

Modified KN smoothing uses a different discount D_r for each count $r < 3$, and a discount D_{3+} for counts $r \geq 3$. On count distributions, a similar argument to the above leads to the estimates:

$$\begin{aligned} D_1 &\leq 1 - 2Y\frac{E[n_2]}{E[n_1]} \\ D_2 &\leq 2 - 3Y\frac{E[n_3]}{E[n_2]} \\ D_{3+} &\approx 3 - 4Y\frac{E[n_4]}{E[n_3]} \\ Y &= \frac{E[n_1]}{E[n_1] + 2E[n_2]}. \end{aligned} \tag{13}$$

One side-effect of this change is that (6) is no longer the correct solution to the marginal constraint (Teh, 2006; Sundermeyer et al., 2011). Although this problem can be fixed, standard implementations simply use (6).

4.5.3 Recursive smoothing

In the original KN method, the lower-order model p' was estimated using (6); recursive KN smoothing applies KN smoothing to p'. To do this, we need to reconstruct counts whose MLE is (6). On integral counts, this is simple: we generate, for each n-gram type $v\mathbf{u}'w$, an $(n-1)$-gram token $\mathbf{u}'w$, for a total of $n_{1+}(\bullet\mathbf{u}'w)$ tokens. We then apply KN smoothing to these counts.

Analogously, on count distributions, for each n-gram type $v\mathbf{u}'w$, we generate an $(n-1)$-gram token $\mathbf{u}'w$ with probability $p(c(v\mathbf{u}'w) > 0)$. Since

$$E[c(\mathbf{u}'w)] = \sum_v p(c(v\mathbf{u}'w) > 0) = E[n_{1+}(\bullet\mathbf{u}'w)],$$

this has (11) as its MLE and therefore satisfies the marginal constraint. We then apply expected KN smoothing to these count distributions.

For the example above, the count distributions used for the unigram distribution would be:

	$r=0$	$r=1$
$p(c(\text{cat}) = r)$	0.14	0.86
$p(c(\text{dog}) = r)$	0.1	0.9

4.6 Summary

In summary, to perform expected KN smoothing (either the original version or Chen and Goodman's modified version), we perform the steps listed below:

	orig.	mod.
compute count distributions	§3.2	
estimate discount D	(10)	(13)
estimate lower-order model p'	(11)	§4.5.3
compute smoothed counts $\tilde{c}$	(8)	(12)
compute probabilities p	(9)	

The computational complexity of expected KN is almost identical to KN on integral counts. The main addition is computing and storing the count distributions. Using the dynamic program in Section 3.2, computing the distributions for each r is linear in the number of n-gram types, and we only need to compute the distributions up to $r = 2$ (or $r = 4$ for modified KN), and store them for $r = 0$ (or up to $r = 2$ for modified KN).

5 Related Work

Witten-Bell (WB) smoothing is somewhat easier than KN to adapt to fractional counts. The SRILM toolkit (Stolcke, 2002) implements a method which we call *fractional WB*:

$$p(w \mid \mathbf{u}) = \lambda(\mathbf{u})p_{\text{mle}}(w \mid \mathbf{u}) + (1 - \lambda(\mathbf{u}))p'(w \mid \mathbf{u}')$$
$$\lambda(\mathbf{u}) = \frac{E[c(\mathbf{u})]}{E[c(\mathbf{u})] + n_{1+}(\mathbf{u}\bullet)},$$

where $n_{1+}(\mathbf{u}\bullet)$ is the number of word types observed after context $\mathbf{u}$, computed by ignoring all weights. This method, although simple, inconsistently uses weights for counting tokens but not types. Moreover, as we will see below, it does not perform as well as expected KN.

The only previous adaptation of KN smoothing to fractional counts that we are aware of is that of Tam and Schultz (2008) and Bisani and Ney (2008), called *fractional KN*. This method subtracts D directly from the fractional counts, zeroing out counts that are smaller than D. The discount D must be set by minimizing an error metric on held-out data using a line search (Tam, p. c.) or Powell's method (Bisani and Ney, 2008), requiring repeated estimation and evaluation of the language model. By contrast, we choose D by leaving-one-out. Like KN on integral counts, our method has a closed-form approximation and requires neither held-out data nor trial and error.

6 Language model adaptation

N-gram language models are widely used in applications like machine translation and speech recognition to select fluent output sentences. Although they can easily be trained on large amounts of data, in order to perform well, they should be trained on data containing the right kind of language. For example, if we want to model spoken language, then we should train on spoken language data. If we train on newswire, then a spoken sentence might be regarded as ill-formed, because the distribution of sentences in these two domains are very different. In practice, we often have limited-size training data from a specific domain, and large amounts of data consisting of language from a variety of domains (we call this *general-domain* data). How can we utilize the large general-domain dataset to help us train a model on a specific domain?

Many methods (Lin et al., 1997; Gao et al., 2002; Klakow, 2000; Moore and Lewis, 2010; Axelrod et al., 2011) rank sentences in the general-domain data according to their similarity to the in-domain data and select only those with score higher than some threshold. Such methods are effective and widely used. However, sometimes it is hard to say whether a sentence is totally in-domain or out-of-domain; for example, quoted speech in a news report might be partly in-domain if the domain of interest is broadcast conversation. Here, we propose to assign each sentence a probability to indicate how likely it is to belong to the domain of interest, and train a language model using expected KN smoothing. We show that this approach yields models with much better perplexity than the original sentence-selection approach.

6.1 Method

One of the most widely used sentence-selection approaches is that of Moore and Lewis (2010). They first train two language models, p_{in} on a set of in-domain data, and p_{out} on a set of general-domain data. Then each sentence $\mathbf{w}$ is assigned a score

$$H(\mathbf{w}) = \frac{\log(p_{\text{in}}(\mathbf{w})) - \log(p_{\text{out}}(\mathbf{w}))}{|\mathbf{w}|}.$$

They set a threshold on the score to select a subset.

We adapt this approach as follows. After selection, for each sentence in the subset, we use a sigmoid function to map the scores into probabilities:

$$p(\mathbf{w} \text{ is in-domain}) = \frac{1}{1 + \exp(-H(\mathbf{w}))}.$$

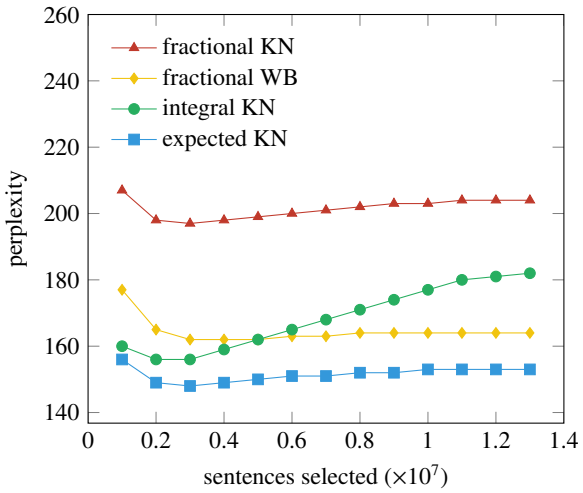


Figure 1: On the language model adaptation task, expected KN outperforms all other methods across all sizes of selected subsets. Integral KN is applied to unweighted instances, while fractional WB, fractional KN and expected KN are applied to weighted instances.

Then we use the weighted subset to train a language model with expected KN smoothing.

6.2 Experiments

Moore and Lewis (2010) test their method by partitioning the in-domain data into training data and test data, both of which are disjoint from the general-domain data. They use the in-domain training data to select a subset of the general-domain data, build a language model on the selected subset, and evaluate its perplexity on the in-domain test data. Here, we follow this experimental framework and compare Moore and Lewis's unweighted method to our weighted method.

For our experiments, we used all the English data allowed for the BOLT Phase 1 Chinese-English evaluation. We took 60k sentences (1.7M words) of web forum data as in-domain data, further subdividing it into 54k sentences (1.5M words) for training, 3k sentences (100k words) for testing, and 3k sentences (100k words) for future use. The remaining 12.7M sentences (268M words) we treated as general-domain data.

We trained trigram language models and compared expected KN smoothing against integral KN smoothing, fractional WB smoothing, and fractional KN smoothing, measuring perplexity across various subset sizes (Figure 1). For fractional KN, for each subset size, we optimized D to mini-

mize perplexity on the *test* set to give it the greatest possible advantage; nevertheless, it is clearly the worst performer. Expected KN consistently gives the best perplexity, and, at the optimal subset size, obtains better perplexity (148) than the other methods (156 for integral KN, 162 for fractional WB and 197 for fractional KN). Finally, we note that integral KN is very sensitive to the subset size, whereas expected KN and the other methods are more robust.

7 Word Alignment

In this section, we show how to apply expected KN to the IBM word alignment models (Brown et al., 1993). This illustrates both how to use expected KN inside EM and how to use it beyond language modeling. Of course, expected KN can be applied to other instances of EM besides word alignment.

7.1 Problem

Given a French sentence $\mathbf{f} = f_1 f_2 \cdots f_m$ and its English translation $\mathbf{e} = e_1 e_2 \cdots e_n$, an alignment $\mathbf{a}$ is a sequence $a_1, a_2, \ldots, a_m$, where a_i is the index of the English word which generates the French word f_i, or NULL. As is common, we assume that each French word can only be generated from one English word or from NULL (Brown et al., 1993; Och and Ney, 2003; Vogel et al., 1996).

The IBM models and related models define probability distributions $p(\mathbf{a}, \mathbf{f} \mid \mathbf{e}, \theta)$, which model how likely a French sentence $\mathbf{f}$ is to be generated from an English sentence $\mathbf{e}$ with word alignment $\mathbf{a}$. Different models parameterize this probability distribution in different ways. For example, Model 1 only models the lexical translation probabilities:

$$p(\mathbf{a}, \mathbf{f} \mid \mathbf{e}, \theta) \propto \prod_{j=1}^{m} p(f_j \mid e_{a_j}).$$

Models 2–5 and the HMM model introduce additional components to model word order and fertility. All, however, have the lexical translation model $p(f_j \mid e_i)$ in common. It also contains most of the model's parameters and is where overfitting occurs most. Thus, here we only apply KN smoothing to the lexical translation probabilities, leaving the other model components for future work.

7.2 Method

The $\mathbf{f}$ and $\mathbf{e}$ are observed, while $\mathbf{a}$ is a latent variable. Normally, in the E step, we collect expected counts $E[c(e, f)]$ for each e and f. Then, in the M step, we find the parameter values that maximize their likelihood. However, MLE is prone to overfitting, one symptom of which is the "garbage collection" phenomenon where a rare English word is wrongly aligned to many French words.

To reduce overfitting, we use expected KN smoothing during the M step. That is, during the E step, we calculate the distribution of $c(e, f)$ for each e and f, and during the M step, we train a language model on bigrams ef using expected KN smoothing (that is, with $\mathbf{u} = e$ and $w = f$). This gives a smoothed probability estimate for $p(f \mid e)$.

One question that arises is: what distribution to use as the lower-order distribution p'? Following common practice in language modeling, we use the unigram distribution $p(f)$ as the lower-order distribution. We could also use the uniform distribution over word types, or a distribution that assigns zero probability to all known word types. (The latter case is equivalent to a backoff language model, where, since all bigrams are known, the lower-order model is never used.) Below, we compare the performance of all three choices.

7.3 Alignment experiments

We modified GIZA++ (Och and Ney, 2003) to perform expected KN smoothing as described above. Smoothing is enabled or disabled with a command-line switch, making direct comparisons simple. Our implementation is publicly available as open-source software.[1]

We carried out experiments on two language pairs: Arabic to English and Czech to English. For Arabic-English, we used 5.4+4.3 million words of parallel text from the NIST 2009 constrained task,[2] and 346 word-aligned sentence pairs (LDC2006E86) for evaluation. For Czech-English, we used all 2.0+2.2 million words of training data from the WMT 2009 shared task, and 515 word-aligned sentence pairs (Bojar and Prokopová, 2006) for evaluation.

For all methods, we used five iterations of IBM Models 1, 2, and HMM, followed by three iterations of IBM Models 3 and 4. We applied expected KN smoothing to all iterations of all models. We aligned in both the foreign-to-English

[1] `https://github.com/hznlp/giza-kn`

[2] All data was used except for: United Nations proceedings (LDC2004E13), ISI Automatically Extracted Parallel Text (LDC2007E08), and Ummah newswire text (LDC2004T18).

Smoothing	p'	Alignment F1		B	
		Ara-Eng	Cze-Eng	Ara-Eng	Cze-Eng
none (baseline)	–	66.5	67.2	37.0	16.6
variational Bayes	uniform	65.7	65.5	36.5	16.6
fractional WB	unigram	60.1	63.7	–	–
	uniform	60.8	66.5	37.8	16.9
	zero	60.8	65.2	–	–
fractional KN	unigram	67.7	70.2	37.2	16.5
expected KN	unigram	**69.7**	**71.9**	**38.2**	**17.0**
	uniform	69.4	71.3	–	–
	zero	69.2	**71.9**	–	–

Table 1: Expected KN (interpolating with the unigram distribution) consistently outperforms all other methods. For variational Bayes, we followed Riley and Gildea (2012) in setting α to zero (so that the choice of p' is irrelevant). For fractional KN, we chose D to maximize F1 (see Figure 2).

and English-to-foreign directions and then used the *grow-diag-final* method to symmetrize them (Koehn et al., 2003), and evaluated the alignments using F-measure against gold word alignments.

As shown in Table 1, for KN smoothing, interpolation with the unigram distribution performs the best, while for WB smoothing, interestingly, interpolation with the uniform distribution performs the best. The difference can be explained by the way the two smoothing methods estimate p'. Consider again a training example with a word e that occurs nowhere else in the training data. In WB smoothing, $p'(f)$ is the empirical unigram distribution. If $\mathbf{f}$ contains a word that is much more frequent than the correct translation of e, then smoothing may actually encourage the model to wrongly align e with the frequent word. This is much less of a problem in KN smoothing, where p' is estimated from bigram types rather than bigram tokens.

We also compared with variational Bayes (Riley and Gildea, 2012) and fractional KN. Overall, expected KN performs the best. Variational Bayes is not consistent across different language pairs. While fractional KN does beat the baseline for both language pairs, the value of D, which we optimized D to maximize F1, is not consistent across language pairs: as shown in Figure 2, on Arabic-English, a smaller D is better, while for Czech-English, a larger D is better. By contrast, expected KN uses a closed-form expression for D that outperforms the best performance of fractional KN.

Table 2 shows that, if we apply expected KN smoothing to only selected stages of training, adding smoothing always brings an improvement,

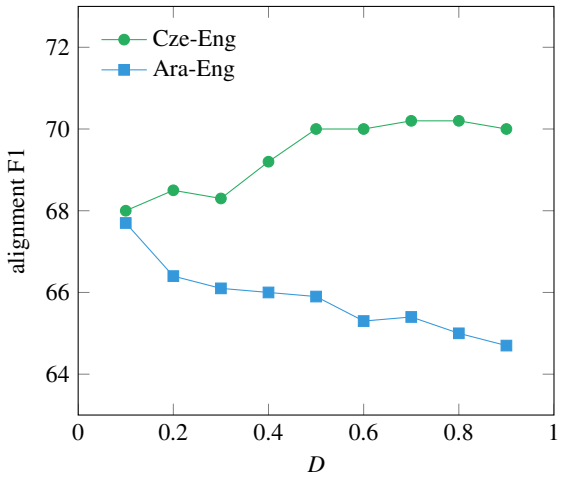


Figure 2: Alignment F1 vs. D of fractional KN smoothing for word alignment.

Smoothed models					Alignment F1	
1	2	H	3	4	Ara-Eng	Cze-Eng
○	○	○	○	○	66.5	67.2
●	○	○	○	○	67.3	67.9
○	●	○	○	○	68.0	68.7
○	○	●	○	○	68.6	70.0
○	○	○	●	○	66.9	68.4
○	○	○	○	●	67.0	68.6
●	●	●	●	●	**69.7**	**71.9**

Table 2: Smoothing more stages of training makes alignment accuracy go up. For each row, we smoothed all iterations of the models indicated. Key: H = HMM model; ● = smoothing enabled; ○ = smoothing disabled.

with the best setting being to smooth all stages. This shows that expected KN smoothing is consistently effective. It is also interesting to note that smoothing is less helpful for the fertility-based Models 3 and 4. Whether this is because modeling fertility makes them less susceptible to "garbage collection," or the way they approximate the E step makes them less amenable to smoothing, or another reason, would require further investigation.

7.4 Translation experiments

Finally, we ran MT experiments to see whether the improved alignments also lead to improved translations. We used the same training data as before. For the Arabic-English tasks, we used the NIST 2008 test set as development data and the NIST 2009 test set as test data; for the Czech-English tasks, we used the WMT 2008 test set as development data and the WMT 2009 test set as test data.

We used the Moses toolkit (Koehn et al., 2007) to build MT systems using various alignments (for expected KN, we used the one interpolated with the unigram distribution, and for fractional WB, we used the one interpolated with the uniform distribution). We used a trigram language model trained on Gigaword (AFP, AP Worldstream, CNA, and Xinhua portions), and minimum error-rate training (Och, 2003) to tune the feature weights.

Table 1 shows that, although the relationship between alignment F1 and B is not very consistent, expected KN smoothing achieves the best B among all these methods and is significantly better than the baseline ($p < 0.01$).

8 Conclusion

For a long time, and as noted by many authors, the usage of KN smoothing has been limited by its restriction to integer counts. In this paper, we addressed this issue by treating fractional counts as distributions over integer counts and generalizing KN smoothing to operate on these distributions. This generalization makes KN smoothing, widely considered to be the best-performing smoothing method, applicable to many new areas. We have demonstrated the effectiveness of our method in two such areas and showed significant improvements in both.

Acknowledgements

We thank Qing Dou, Ashish Vaswani, Wilson Yik-Cheung Tam, and the anonymous reviewers for their input to this work. This research was supported in part by DOI IBC grant D12AP00225.

References

Jesús Andrés-Ferrer. 2010. *Statistical approaches for natural language modelling and monotone statistical machine translation*. Ph.D. thesis, Universidad Politécnica de Valencia.

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proc. EMNLP*, pages 355–362.

Maximilian Bisani and Hermann Ney. 2008. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, 50(5):434–451.

Ondřej Bojar and Magdalena Prokopová. 2006. Czech-English word alignment. In *Proc. LREC*, pages 1236–1239.

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

Stanley F. Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13:359–394.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38.

Jianfeng Gao, Joshua Goodman, Mingjing Li, and Kai-Fu Lee. 2002. Toward a unified approach to statistical language modeling for Chinese. *ACM Transactions on Asian Language Information*, 1:3–33.

Joshua T. Goodman. 2001. A bit of progress in language modeling: Extended version. Technical Report MSR-TR-2001-72, Microsoft Research.

Joshua Goodman. 2004. Exponential priors for maximum entropy models. In *Proc. HLT-NAACL*, pages 305–312.

Yili Hong. 2013. On computing the distribution function for the Poisson binomial distribution. *Computational Statistics and Data Analysis*, 59:41–51.

Dietrich Klakow. 2000. Selecting articles from the language model training corpus. In *Proc. ICASSP*, pages 1695–1698.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for *M*-gram language modeling. In *Proc. ICASSP 1995*, pages 181–184.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. HLT-NAACL*, pages 127–133.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. ACL, Companion Volume*, pages 177–180.

Sung-Chien Lin, Chi-Lung Tsai, Lee-Feng Chien, Keh-Jiann Chen, and Lin-Shan Lee. 1997. Chinese language model adaptation based on document classification and multiple domain-specific language models. In *Proc. Eurospeech*, pages 1463–1466.

Robert Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proc. ACL*, pages 220–224.

Hermann Ney, Ute Essen, and Reinhard Kneser. 1994. On structuring probabilistic dependencies in stochastic language modelling. *Computer Speech and Language*, 8:1–38, 1.

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

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. ACL*, pages 160–167.

Darcey Riley and Daniel Gildea. 2012. Improving the IBM alignment models using variational Bayes. In *Proc. ACL (Volume 2: Short Papers)*, pages 306–310.

Andreas Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Proc. International Conference on Spoken Language Processing*, volume 2, pages 901–904.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2011. On the estimation of discount parameters for language model smoothing. In *Proc. Interspeech*, pages 1433–1436.

Yik-Cheung Tam and Tanja Schultz. 2008. Correlated bigram LSA for unsupervised language model adaptation. In *Proc. NIPS*, pages 1633–1640.

Yee Whye Teh. 2006. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proc. COLING-ACL*, pages 985–992.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proc. COLING*, pages 836–841.

Joern Wuebker, Mei-Yuh Hwang, and Chris Quirk. 2012. Leave-one-out phrase model training for large-scale deployment. In *Proc. WMT*, pages 460–467.

Robust Entity Clustering via Phylogenetic Inference

Nicholas Andrews and **Jason Eisner** and **Mark Dredze**
Department of Computer Science and Human Language Technology Center of Excellence
Johns Hopkins University
3400 N. Charles St., Baltimore, MD 21218 USA
{noa,eisner,mdredze}@jhu.edu

Abstract

Entity clustering must determine when two named-entity mentions refer to the same entity. Typical approaches use a pipeline architecture that clusters the mentions using fixed or learned measures of name and context similarity. In this paper, we propose a model for cross-document coreference resolution that achieves robustness by learning similarity from unlabeled data. The generative process assumes that each entity mention arises from copying and optionally mutating an earlier name from a similar context. Clustering the mentions into entities depends on recovering this copying tree jointly with estimating models of the mutation process and parent selection process. We present a block Gibbs sampler for posterior inference and an empirical evaluation on several datasets.

1 Introduction

Variation poses a serious challenge for determining who or what a name refers to. For instance, Wikipedia contains more than 100 variations of the name Barack Obama as redirects to the U.S. President article, including:

President Obama	Barack H. Obama, Jr.
Barak Obamba	Barry Soetoro

To relate different names, one solution is to use specifically tailored measures of name similarity such as Jaro-Winkler similarity (Winkler, 1999; Cohen et al., 2003). This approach is brittle, however, and fails to adapt to the test data. Another option is to train a model like stochastic edit distance from known pairs of similar names (Ristad and Yianilos, 1998; Green et al., 2012), but this requires supervised data in the test domain.

Even the best model of name similarity is not enough by itself, since two names that are similar—even identical—do not *necessarily* corefer. Document context is needed to determine whether they may be talking about two different people.

In this paper, we propose a method for jointly (1) learning similarity between names and (2) clustering name mentions into entities, the two major components of cross-document coreference resolution systems (Baron and Freedman, 2008; Finin et al., 2009; Rao et al., 2010; Singh et al., 2011; Lee et al., 2012; Green et al., 2012). Our model is an evolutionary generative process based on the name variation model of Andrews et al. (2012), which stipulates that names are often copied from previously generated names, perhaps with mutation (spelling edits). This can deduce that rather than being names for different entities, Barak Obamba and Barock obama more likely arose from the frequent name Barack Obama as a common ancestor, which accounts for most of their letters. This can also relate seemingly dissimilar names via multiple steps in the generative process:

Taylor Swift → T-Swift → T-Swizzle

Our model learns without supervision that these all refer to the the same entity. Such creative spellings are especially common on Twitter and other social media; we give more examples of coreferents learned by our model in Section 8.4.

Our primary contributions are improvements on Andrews et al. (2012) for the entity clustering task. Their inference procedure only clustered types (distinct names) rather than tokens (mentions in context), and relied on expensive matrix inversions for learning. Our novel approach features:

- §4.1 A topical model of which entities from previously written text an author tends to mention from previously written text.
- §4.2 A name mutation model that is sensitive to features of the input and output characters and takes a reader's comprehension into account.
- §5 A scalable Markov chain Monte Carlo sampler used in training and inference.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 775–785,
Baltimore, Maryland, USA, June 23-25 2014.

§7 A minimum Bayes risk decoding procedure to pick an output clustering. The procedure is applicable to any model capable of producing a posterior over coreference decisions.

We evaluate our approach by comparing to several baselines on datasets from three different genres: Twitter, newswire, and blogs.

2 Overview and Related Work

Cross-document coreference resolution (CDCR) was first introduced by Bagga and Baldwin (1998b). Most approaches since then are based on the intuitions that coreferent names tend to have "similar" spellings and tend to appear in "similar" contexts. The distinguishing feature of our system is that both notions of similarity are learned together without supervision.

We adopt a "phylogenetic" generative model of coreference. The basic insight is that coreference is created when an author thinks of an entity that was mentioned earlier in a similar context, and mentions it again in a similar way. The author may alter the name mention string when copying it, but both names refer to the same entity. Either name may later be copied further, leading to an evolutionary tree of mentions—a phylogeny. Phylogenetic models are new to information extraction. In computational historical linguistics, Bouchard-Côté et al. (2013) have also modeled the mutation of strings along the edges of a phylogeny; but for them the phylogeny is observed and most mentions are not, while we observe the mentions only.

To apply our model to the CDCR task, we observe that the probability that two name mentions are coreferent is the probability that they arose from a common ancestor in the phylogeny. So we design a Monte Carlo sampler to reconstruct likely phylogenies. A phylogeny must explain every observed name. While our model is capable of generating each name independently, *a phylogeny will generally achieve higher probability if it explains similar names as being similar by mutation (rather than by coincidence)*. Thus, our sampled phylogenies tend to make similar names coreferent—especially long or unusual names that would be expensive to generate repeatedly, and especially in contexts that are topically similar and therefore have a higher prior probability of coreference.

For learning, we iteratively adjust our model's parameters to better explain our samples. That is, we do unsupervised training via Monte Carlo EM.

What is learned? An important component of a CDCR system is its model of name similarity (Winkler, 1999; Porter and Winkler, 1997), which is often fixed up front. This role is played in our system by the name mutation model, which we take to be a variant of stochastic edit distance (Ristad and Yianilos, 1996). Rather than fixing its parameters before we begin CDCR, we *learn* them (without supervision) as part of CDCR, by training from samples of reconstructed phylogenies.

Name similarity is also an important component of *within*-document coreference resolution, and efforts in that area bear resemblance to our approach. Haghighi and Klein (2010) describe an "entity-centered" model where a distance-dependent Chinese restaurant process is used to pick previous coreferent mentions *within* a document. Similarly, Durrett and Klein (2013) learn a mention similarity model based on labeled data. Our *cross*-document setting has no observed mention ordering and no observed entities: we must sum over all possibilities, a challenging inference problem.

The second major component of CDCR is context-based disambiguation of similar or identical names that refer to the same entity. Like Kozareva and Ravi (2011) and Green et al. (2012) we use topics as the contexts, but learn mention topics *jointly* with other model parameters.

3 Generative Model of Coreference

Let $\boldsymbol{x} = (x_1, \ldots, x_N)$ denote an ordered sequence of distinct named-entity mentions in documents $\boldsymbol{d} = (d_1, \ldots, d_D)$. We assume that each document has a (single) known language, and that its mentions and their types have been identified by a named-entity recognizer. We use the object-oriented notation $x.v$ for attribute v of mention x.

Our model generates an ordered sequence $\boldsymbol{x}$ although we do not observe its order. Thus each mention x has *latent* position $x.i$ (e.g., $x_{729}.i = 729$). The entire corpus, including these entities, is generated according to standard topic model assumptions; we first generate a topic distribution for a document, then sample topics and words for the document (Blei et al., 2003). However, any topic may generate an entity type, e.g. PERSON, which is then replaced by a specific name: when PERSON is generated, the model chooses a *previous* mention of any person and copies it, perhaps mutating its name.[1] Alternatively, the model may manufacture

[1] We make the closed-world assumption that the author is

a name for a new person, though the name itself may not be new.

If all previous *mentions* were equally likely, this would be a Chinese Restaurant Process (CRP) in which frequently mentioned *entities* are more likely to be mentioned again ("the rich get richer"). We refine that idea by saying that the current topic, language, and document influence the choice of which previous mention to copy, similar to the distance-dependent CRP (Blei and Frazier, 2011).[2] This will help distinguish multiple John Smith entities if they tend to appear in different contexts.

Formally, each mention x is derived from a parent mention $x.p$ where $x.p.i < x.i$ (the parent came first), $x.e = x.p.e$ (same entity) and $x.n$ is a copy or mutation of $x.p.n$. In the special case where x is a first mention of $x.e$, $x.p$ is the special symbol $\diamondsuit$, $x.e$ is a newly allocated entity of some appropriate type, and the name $x.n$ is generated from scratch.

Our goal is to reconstruct mappings $\boldsymbol{p}, \boldsymbol{i}, \boldsymbol{z}$ that specify the latent properties of the mentions x. The mapping $\boldsymbol{p} : x \mapsto x.p$ forms a phylogenetic tree on the mentions, with root $\diamondsuit$. Each entity corresponds to a subtree that is rooted at some child of $\diamondsuit$. The mapping $\boldsymbol{i} : x \mapsto x.i$ gives an ordering consistent with that tree in the sense that $(\forall x) x.p.i < x.i$. Finally, the mapping $\boldsymbol{z} : x \mapsto x.z$ specifies, for each mention, the topic that generated it. While $\boldsymbol{i}$ and $\boldsymbol{z}$ are not necessary for creating coref clusters, they are needed to produce $\boldsymbol{p}$.

4 Detailed generative story

Given a few constants that are referenced in the main text, we assume that the corpus $\boldsymbol{d}$ was generated as follows.

First, for each topic $z = 1, \ldots K$ and each language ℓ, choose a multinomial $\boldsymbol{\beta}_{z\ell}$ over the word vocabulary, from a symmetric Dirichlet with concentration parameter η. Then set $m = 0$ (entity count), $i = 0$ (mention count), and for each document index $d = 1, \ldots, D$:

1. Choose the document's length L and language ℓ. (The distributions used to choose these are unimportant because these variables are always observed.)
2. Choose its topic distribution $\boldsymbol{\psi}_d$ from an asymmetric Dirichlet prior with parameters $\boldsymbol{m}$ (Wallach et al., 2009).[3]
3. For each token position $k = 1, \ldots, L$:
   (a) Choose a topic $z_{dk} \sim \boldsymbol{\psi}_d$.
   (b) Choose a word conditioned on the topic and language, $w_{dk} \sim \boldsymbol{\beta}_{z_{dk}\ell}$.
   (c) If w_{dk} is a named entity type (PERSON, PLACE, ORG, ...) rather than an ordinary word, then increment i and:
      i. create a new mention x with
      $x.e.t = w_{dk} \quad x.d = d \quad x.\ell = \ell$
      $x.i = i \quad x.z = z_{dk} \quad x.k = k$
      ii. Choose the parent $x.p$ from a distribution conditioned on the attributes just set (see §4.1).
      iii. If $x.p = \diamondsuit$, increment m and set $x.e =$ a new entity e_m. Else set $x.e = x.p.e$.
      iv. Choose $x.n$ from a distribution conditioned on $x.p.n$ and $x.\ell$ (see §4.2).

Notice that the tokens w_{dk} in document d are exchangeable: by collapsing out $\boldsymbol{\psi}_d$, we can regard them as having been generated from a CRP. Thus, for fixed values of the non-mention tokens and their topics, the probability of generating the mention sequence $\boldsymbol{x}$ is proportional to the product of the probabilities of the choices in step 3 at the positions dk where mentions were generated. These choices generate a topic $x.z$ (from the CRP for document d), a type $x.e.t$ (from $\beta_{x.z}$), a parent mention (from the distribution over previous mentions), and a name string (conditioned on the parent's name if any). §5 uses this fact to construct an MCMC sampler for the latent parts of $\boldsymbol{x}$.

4.1 Sub-model for parent selection

To select a parent for a mention x of type $t = x.e.t$, a simple model (as mentioned above) would be a CRP: each previous mention of the same type is selected with probability proportional to 1, and $\diamondsuit$ is

only aware of previous mentions *from our corpus*. This means that two mentions cannot be derived from a common ancestor outside our corpus. To mitigate this unrealistic assumption, we allow any ordering $\boldsymbol{x}$ of the observed mentions, not respecting document timestamps or forcing the mentions from a given document to be generated as a contiguous subsequence of $\boldsymbol{x}$.

[2] Unlike the ddCRP, our generative story is careful to prohibit derivational cycles: each mention is copied from a *previous* mention in the latent ordering. This is why our phylogeny is a *tree*, and why our sampler is more complex. Also unlike the ddCRP, we permit asymmetric "distances": if a certain topic or language likes to copy mentions from another, the compliment is not necessarily returned.

[3] Extension: This choice could depend on the language $d.\ell$.

selected with probability proportional to $\alpha_t > 0$. A larger choice of α_t results in smaller entity clusters, because it prefers to create new entities of type t rather than copying old ones.

We modify this story by re-weighting $\diamondsuit$ and previous mentions according to their relative suitability as the parent of x:

$$\Pr_{\boldsymbol{\phi}}(x.p \mid x) = \frac{\exp\left(\boldsymbol{\phi} \cdot \mathbf{f}(x.p, x)\right)}{Z(x)} \quad (1)$$

where $x.p$ ranges over $\diamondsuit$ and all previous mentions of the same type as x, that is, mentions p such that $p.i < x.i$ and $p.e.t = x.e.t$. The normalizing constant $Z(x) \stackrel{\text{def}}{=} \sum_p \exp\left(\boldsymbol{\phi} \cdot \mathbf{f}(x.p, x)\right)$ is chosen so that the probabilities sum to 1.

This is a conditional log-linear model parameterized by $\boldsymbol{\phi}$, where $\phi_k \sim \mathcal{N}(0, \sigma_k^2)$. The features $\mathbf{f}$ are extracted from the attributes of x and $x.p$. Our most important feature tests whether $x.p.z = x.z$. This binary feature has a high weight if authors mainly choose mentions from the same topic. To model which (other) topics tend to be selected, we also have a binary feature for each parent topic $x.p.z$ and each topic pair $(x.p.z, x.z)$.[4]

4.2 Sub-model for name mutation

Let x denote a mention with parent $p = x.p$. As in Andrews et al. (2012), its name $x.n$ is a stochastic transduction of its parent's name $p.n$. That is,

$$\Pr_{\boldsymbol{\theta}}(x.n \mid p.n) \quad (2)$$

is given by the probability that applying a random sequence of edits to the characters of $p.n$ would yield $x.n$. The contextual probabilities of different edits depend on learned parameters $\boldsymbol{\theta}$.

(2) is the total probability of *all* edit sequences that derive $x.n$ from $p.n$. It can be computed in time $O(|x.n| \cdot |p.n|)$ by dynamic programming.

The probability of a *single* edit sequence, which corresponds to a monotonic alignment of $x.n$ to $p.n$, is a product of individual edit probabilities of the form $\Pr_{\boldsymbol{\theta}}(\binom{a}{b} \mid \hat{a})$, which is conditioned on the next input character $\hat{a}$. The edit $\binom{a}{b}$ replaces input $a \in \{\epsilon, \hat{a}\}$ with output $b \in \{\epsilon\} \cup \Sigma$ (where ϵ is the empty string and Σ is the alphabet of language $x.\ell$). Insertions and deletions are the cases where respectively $a = \epsilon$ or $b = \epsilon$—we do not allow both at once. All other edits are substitutions. When $\hat{a}$ is the special end-of-string symbol $\#$, the only allowed edits are the insertion $\binom{\epsilon}{b}$ and the substitution $\binom{\#}{\#}$. We define the edit probability using a locally normalized log-linear model:

$$\Pr_{\boldsymbol{\theta}}(\binom{a}{b} \mid \hat{a}) = \frac{\exp(\boldsymbol{\theta} \cdot \mathbf{f}(\hat{a}, a, b))}{\sum_{a', b'} \exp(\boldsymbol{\theta} \cdot \mathbf{f}(\hat{a}, a', b'))} \quad (3)$$

We use a small set of simple feature functions $\mathbf{f}$, which consider conjunctions of the attributes of the characters $\hat{a}$ and b: character, character class (letter, digit, etc.), and case (upper vs. lower).

More generally, the probability (2) may also be conditioned on other variables such as on the languages $p.\ell$ and $x.\ell$—this leaves room for a transliteration model when $x.\ell \neq p.\ell$—and on the entity type $x.t$. The features in (3) may then depend on these variables as well.

Notice that we use a locally normalized probability for each edit. This enables faster and simpler training than the similar model of Dreyer et al. (2008), which uses a globally normalized probability for the whole edit sequence.

When $p = \diamondsuit$, we are generating a new name $x.n$. We use the same model, taking $\diamondsuit.n$ to be the empty string (but with $\#_\diamondsuit$ rather than $\#$ as the end-of-string symbol). This yields a feature-based unigram language model (whose character probabilities may differ from usual insertion probabilities because they see $\#_\diamondsuit$ as the lookahead character).

Pragmatics. We can optionally make the model more sophisticated. Authors tend to *avoid* names $x.n$ that readers would misinterpret (given the previously generated names). The edit model thinks that $\Pr_{\boldsymbol{\theta}}(\textsf{CIA} \mid \diamondsuit)$ is relatively high (because CIA is a short string) and so is $\Pr_{\boldsymbol{\theta}}(\textsf{CIA} \mid \textsf{Chuck's Ice Art})$. But in fact, if CIA has already been frequently used to refer to the Central Intelligence Agency, then an author is unlikely to use it for a different entity.

To model this pragmatic effect, we multiply our definition of $\Pr_{\boldsymbol{\theta}}(x.n \mid p.n)$ by an extra factor $\Pr(x.e \mid x)^\gamma$, where $\gamma \geq 0$ is the effect strength.[5] Here $\Pr(x.e \mid x)$ is the probability that a reader correctly identifies the entity $x.e$. We take this to be the probability that a reader who knows our sub-models would guess some parent

[4] Many other features could be added. In a multilingual setting, one would similarly want to model whether English authors select Arabic mentions. One could also imagine features that reward proximity in the generative order ($x.p.i \approx x.i$), local linguistic relationships (when $x.p.d = x.d$ and $x.p.k \approx x.k$), or social information flow (e.g., from mainstream media to Twitter). One could also make more specific versions of any feature by conjoining it with the entity type t.

[5] Currently we omit the step of renormalizing this deficient model. Our training procedure also ignores the extra factor.

having the correct entity (or $\diamondsuit$ if x is a first mention): $\sum_{p':p'.e=x.e} w(p', x) / \sum_{p'} w(p', x)$. Here p' ranges over mentions (including $\diamondsuit$) that precede x in the ordering $\boldsymbol{i}$, and $w(p', x)$—defined later in sec. 5.3—is proportional to the posterior probability that $x.p = p'$, given name $x.n$ and topic $x.z$.[6]

5 Inference by Block Gibbs Sampling

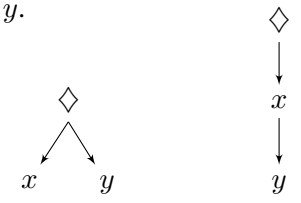
We use a block Gibbs sampler, which from an initial state $(\boldsymbol{p}_0, \boldsymbol{i}_0, \boldsymbol{z}_0)$ repeats these steps:

1. Sample the ordering $\boldsymbol{i}$ from its conditional distribution given all other variables.
2. Sample the topic vector $\boldsymbol{z}$ likewise.
3. Sample the phylogeny $\boldsymbol{p}$ likewise.
4. Output the current sample $s_t = (\boldsymbol{p}, \boldsymbol{i}, \boldsymbol{z})$.

It is difficult to draw exact samples at steps 1 and 2. Thus, we sample $\boldsymbol{i}$ or $\boldsymbol{z}$ from a simpler proposal distribution, but correct the discrepancy using the Independent Metropolis-Hastings (IMH) strategy: with an appropriate probability, reject the proposed new value and instead use another copy of the current value (Tierney, 1994).

5.1 Resampling the ordering i

We resample the ordering $\boldsymbol{i}$ of the mentions $\boldsymbol{x}$, conditioned on the other variables. The current phylogeny $\boldsymbol{p}$ already defines a partial order on $\boldsymbol{x}$, since each parent must precede its children. For instance, phylogeny (a) below requires $\diamondsuit \prec x$ and $\diamondsuit \prec y$. This partial order is compatible with 2 total orderings, $\diamondsuit \prec x \prec y$ and $\diamondsuit \prec y \prec x$. By contrast, phylogeny (b) requires the total ordering $\diamondsuit \prec x \prec y$.



We first sample an ordering $\boldsymbol{i}_\diamondsuit$ (the ordering of mentions with parent $\diamondsuit$, i.e. all mentions) *uniformly* at random from the set of orderings compatible with the current $\boldsymbol{p}$. (We provide details about this procedure in Appendix A.)[7] However, such orderings are not in fact equiprobable given the other variables—some orderings better explain why that phylogeny was chosen in the first place, according to our competitive parent selection model (§4.1). To correct for this bias using IMH, we accept the proposed ordering $\boldsymbol{i}_\diamondsuit$ with probability

$$a = \min\left(1, \frac{\Pr(\boldsymbol{p}, \boldsymbol{i}_\diamondsuit, \boldsymbol{z}, \boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{\phi})}{\Pr(\boldsymbol{p}, \boldsymbol{i}, \ \boldsymbol{z}, \boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{\phi})}\right) \quad (4)$$

where $\boldsymbol{i}$ is the current ordering. Otherwise we reject $\boldsymbol{i}_\diamondsuit$ and reuse $\boldsymbol{i}$ for the new sample.

5.2 Resampling the topics z

Each context word and each named entity is associated with a latent topic. The topics of *context words* are assumed exchangeable, and so we resample them using Gibbs sampling (Griffiths and Steyvers, 2004).

Unfortunately, this is prohibitively expensive for the (non-exchangeable) topics of the *named mentions* x. A Gibbs sampler would have to choose a new value for $x.z$ with probability proportional to the resulting joint probability of the full sample. This probability is expensive to evaluate because changing $x.z$ will change the probability of *many* edges in the current phylogeny $\boldsymbol{p}$. (Equation (1) puts x is in competition with other parents, so *every* mention y that follows x must recompute how happy it is with its current parent $y.p$.)

Rather than resampling one topic at a time, we resample $\boldsymbol{z}$ as a block. We use a proposal distribution for which block sampling is efficient, and use IMH to correct the error in this proposal distribution.

Our proposal distribution is an undirected graphical model whose random variables are the topics $\boldsymbol{z}$ and whose graph structure is given by the current phylogeny $\boldsymbol{p}$:

$$Q(\boldsymbol{z}) \propto \prod_{x \neq \diamondsuit} \Psi_x(x.z) \Psi_{x.p,x}(x.p.z, x.z) \quad (5)$$

$Q(\boldsymbol{z})$ is an approximation to the posterior distribution over $\boldsymbol{z}$. As detailed below, a proposal can be sampled from $Q(\boldsymbol{z})$ in time $O(|\boldsymbol{z}|K^2)$ where K is the number of topics, because the only interactions among topics are along the edges of the tree $\boldsymbol{p}$. The unary factor Ψ_x gives a weight for each possible value of $x.z$, and the binary factor $\Psi_{x.p,x}$ gives a weight for each possible value of the pair $(x.p.z, x.z)$.

The $\Psi_x(x.z)$ factors in (5) approximate the topic model's prior distribution over $\boldsymbol{z}$. $\Psi_x(x.z)$ is proportional to the probability that a Gibbs sampling step for an ordinary topic model would choose this value of $x.z$. This depends on whether—in the

[6] Better, one could integrate over the reader's *guess* of $x.z$.

[7] The full version of this paper is available at `http://cs.jhu.edu/~noa/publications/phylo-acl-14.pdf`

current sample—$x.z$ is currently common in x's document and $x.t$ is commonly generated by $x.z$. It ignores the fact that we will also be resampling the topics of the other mentions.

The $\Psi_{x.p,x}$ factors in (5) approximate $\Pr(\boldsymbol{p} \mid \boldsymbol{z}, \boldsymbol{i})$ (up to a constant factor), where $\boldsymbol{p}$ is the current phylogeny. Specifically, $\Psi_{x.p,x}$ approximates the probability of a single edge. It ought to be given by (1), but we use only the numerator of (1), which avoids modeling the competition among parents.

We sample from Q using standard methods, similar to sampling from a linear-chain CRF by running the backward algorithm followed by forward sampling. Specifically, we run the sum-product algorithm from the leaves up to the root $\diamondsuit$, at each node x computing the following for each topic z:

$$\beta_x(z) \stackrel{\text{def}}{=} \Psi_x(z) \cdot \prod_{y \in \text{children}(x)} \sum_{z'} \Psi_{x,y}(z, z') \cdot \beta_y(z')$$

Then we sample from the root down to the leaves, first sampling $\diamondsuit.z$ from $\beta_\diamondsuit$, then at each $x \neq \diamondsuit$ sampling the topic $x.z$ to be z with probability proportional to $\Psi_{x.p,x}(x.p.z, z) \cdot \beta_x(z)$.

Again we use IMH to correct for the bias in Q: we accept the resulting proposal $\hat{\boldsymbol{z}}$ with probability

$$\min\left(1, \frac{\Pr(\boldsymbol{p}, \boldsymbol{i}, \hat{\boldsymbol{z}}, \boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{\phi})}{\Pr(\boldsymbol{p}, \boldsymbol{i}, \boldsymbol{z}, \boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{\phi})} \cdot \frac{Q(\boldsymbol{z})}{Q(\hat{\boldsymbol{z}})}\right) \quad (6)$$

While $\Pr(\boldsymbol{p}, \boldsymbol{i}, \hat{\boldsymbol{z}}, \boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{\phi})$ might seem slow to compute because it contains many factors (1) with different denominators $Z(x)$, one can share work by visiting the mentions x in their order $\boldsymbol{i}$. Most summands in $Z(x)$ were already included in $Z(x')$, where x' is the latest previous mention having the same attributes as x (e.g., same topic).

5.3 Resampling the phylogeny $\boldsymbol{p}$

It is easy to resample the phylogeny. For each x, we must choose a parent $x.p$ from among the possible parents p (having $p.i < x.i$ and $p.e.t = x.e.t$). Since the ordering $\boldsymbol{i}$ prevents cycles, the resulting phylogeny $\boldsymbol{p}$ is indeed a tree.

Given the topics $\boldsymbol{z}$, the ordering $\boldsymbol{i}$, and the observed names, we choose an $x.p$ value according to its posterior probability. This is proportional to $w(x.p, x) \stackrel{\text{def}}{=} \Pr_{\boldsymbol{\phi}}(x.p \mid x) \cdot \Pr_{\boldsymbol{\theta}}(x.n \mid x.p.n)$, independent of any other mention's choice of parent. The two factors here are given by (1) and (2) respectively. As in the previous section, the denominators $Z(x)$ in the $\Pr(x.p \mid x)$ factors can be computed efficiently with shared work.

With the pragmatic model (section 4.2), the parent choices are no longer independent; then the samples of $\boldsymbol{p}$ should be corrected by IMH as usual.

5.4 Initializing the sampler

The initial sampler state $(\boldsymbol{z}_0, \boldsymbol{p}_0, \boldsymbol{i}_0)$ is obtained as follows. (1) We fix topics $\boldsymbol{z}_0$ via collapsed Gibbs sampling (Griffiths and Steyvers, 2004). The sampler is run for 1000 iterations, and the final sampler state is taken to be $\boldsymbol{z}_0$. This process treats all topics as exchangeable, including those associated with named entities.(2) Given the topic assignment $\boldsymbol{z}_0$, initialize $\boldsymbol{p}_0$ to the phylogeny rooted at $\diamondsuit$ that maximizes $\sum_x \log w(x.p, x)$. This is a maximum rooted directed spanning tree problem that can be solved in time $O(n^2)$ (Tarjan, 1977). The weight $w(x.p, x)$ is defined as in section 5.3—except that since we do not yet have an ordering $\boldsymbol{i}$, we do not restrict the possible values of $x.p$ to mentions p with $p.i < x.p.i$. (3) Given $\boldsymbol{p}_0$, sample an ordering $\boldsymbol{i}_0$ using the procedure described in §5.1.

6 Parameter Estimation

Evaluating the likelihood and its partial derivatives with respect to the parameters of the model requires marginalizing over our latent variables. As this marginalization is intractable, we resort to Monte Carlo EM procedure (Levine and Casella, 2001) which iterates the following two steps:

E-step: Collect samples by MCMC simulation as in §5, given current model parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$.

M-step: Improve $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ to increase[8]

$$\mathcal{L} \stackrel{\text{def}}{=} \frac{1}{S} \sum_{s=1}^{S} \log \Pr_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{p}_s, \boldsymbol{i}_s, \boldsymbol{z}_s) \quad (7)$$

It is not necessary to locally maximize $\mathcal{L}$ at each M-step, merely to improve it if it is not already at a local maximum (Dempster et al., 1977). We improve it by a single update: at the tth M-step, we update our parameters to $\Phi_t = (\boldsymbol{\theta}_t, \boldsymbol{\phi}_t)$

$$\Phi_t = \Phi_{t-1} + \varepsilon \Sigma_t \nabla_\Phi \mathcal{L}(\boldsymbol{x}, \Phi_{t-1}) \quad (8)$$

where ε is a fixed scaling term and Σ_t is an adaptive learning rate given by AdaGrad (Duchi et al., 2011).

We now describe how to compute the gradient $\nabla_\Phi \mathcal{L}$. The gradient with respect to the parent se-

[8] We actually do MAP-EM, which augments (7) by adding the log-likelihoods of θ and ϕ under a Gaussian prior.

lection parameters $\boldsymbol{\phi}$ is

$$\sum \frac{1}{S} \left(\mathbf{f}(p, x) - \sum_{p'} \Pr_{\boldsymbol{\phi}}(p' \mid x) \mathbf{f}(p', x) \right) \quad (9)$$

The outer summation ranges over all edges in the S samples. The other variables in (9) are associated with the edge being summed over. That edge explains a mention x as a mutation of some parent p in the context of a particular sample $(\boldsymbol{p}_s, \boldsymbol{i}_s, \boldsymbol{z}_s)$. The possible parents p' range over $\diamondsuit$ and the mentions that precede x according to the ordering $\boldsymbol{i}_s$, while the features $\boldsymbol{f}$ and distribution $\Pr_{\boldsymbol{\phi}}$ depend on the topics $\boldsymbol{z}_s$.

As for the mutation parameters, let $c_{p,x}$ be the fraction of samples in which p is the parent of x. This is the expected number of times that the string $p.n$ mutated into $x.n$. Given this weighted set of string pairs, let $c_{\hat{a},a,b}$ be the expected number of times that edit $\binom{a}{b}$ was chosen in context $\hat{a}$: this can be computed using dynamic programming to marginalize over the latent edit sequence that maps $p.n$ to $x.n$, for each (p, x). The gradient of $\mathcal{L}$ with respect to $\boldsymbol{\theta}$ is

$$\sum_{\hat{a},a,b} c_{\hat{a},a,b} (\mathbf{f}(\hat{a}, a, b) - \sum_{a',b'} \Pr_{\boldsymbol{\theta}}(a', b' \mid \hat{a}) \mathbf{f}(\hat{a}, a', b')) \quad (10)$$

7 Consensus Clustering

From a single phylogeny $\boldsymbol{p}$, we deterministically obtain a clustering $\boldsymbol{e}$ by removing the root $\diamondsuit$. Each of the resulting connected components corresponds to a cluster of mentions. Our model gives a distribution over phylogenies $\boldsymbol{p}$ (given observations $\boldsymbol{x}$ and learned parameters Φ)—and thus gives a posterior distribution over clusterings $\boldsymbol{e}$, which can be used to answer various queries.

A traditional query is to request a *single* clustering $\boldsymbol{e}$. We prefer the clustering $\boldsymbol{e}^*$ that minimizes Bayes risk (MBR) (Bickel and Doksum, 1977):

$$\boldsymbol{e}^* = \operatorname*{argmin}_{\boldsymbol{e}'} \sum_{\boldsymbol{e}} L(\boldsymbol{e}', \boldsymbol{e}) \Pr(\boldsymbol{e} \mid \boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{\phi}) \quad (11)$$

This minimizes our expected loss, where $L(\boldsymbol{e}', \boldsymbol{e})$ denotes the loss associated with picking $\boldsymbol{e}'$ when the true clustering is $\boldsymbol{e}$. In practice, we again estimate the expectation by sampling $\boldsymbol{e}$ values.

The Rand index (Rand, 1971)—unlike our actual evaluation measure—is an *efficient* choice of loss function L for use with (11):

$$R(\boldsymbol{e}', \boldsymbol{e}) \stackrel{\text{def}}{=} \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} = \frac{\text{TP} + \text{TN}}{\binom{N}{2}}$$

where the true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) use the clustering $\boldsymbol{e}$ to evaluate how well $\boldsymbol{e}'$ classifies the $\binom{N}{2}$ mention pairs as coreferent or not. More similar clusterings achieve larger R, with $R(\boldsymbol{e}', \boldsymbol{e}) = 1$ iff $\boldsymbol{e}' = \boldsymbol{e}$. In all cases, $0 \leq R(\boldsymbol{e}', \boldsymbol{e}) = R(\boldsymbol{e}, \boldsymbol{e}') \leq 1$.

The MBR decision rule for the (negated) Rand index is easily seen to be equivalent to

$$\begin{aligned} \boldsymbol{e}^* &= \operatorname*{argmax}_{\boldsymbol{e}'} \mathbb{E}[\text{TP}] + \mathbb{E}[\text{TN}] \quad (12) \\ &= \operatorname*{argmax}_{\boldsymbol{e}'} \sum_{i,j:\, x_i \sim x_j} s_{ij} + \sum_{i,j:\, x_i \not\sim x_j} (1 - s_{ij}) \end{aligned}$$

where $\sim$ denotes coreference according to $\boldsymbol{e}'$. As explained above, the s_{ij} are coreference probabilities s_{ij} that can be estimated from a sample of clusterings $\boldsymbol{e}$.

This objective corresponds to min-max graph cut (Ding et al., 2001), an NP-hard problem with an approximate solution (Nie et al., 2010).[9]

8 Experiments

In this section, we describe experiments on three different datasets. Our main results are described first: Twitter features many instances of name variation that we would like our model to be able to learn. We also report the performance of different ablations of our full approach, in order to see which consistently helped across the different splits. We report additional experiments on the ACE 2008 corpus, and on a political blog corpus, to demonstrate that our approach is applicable in different settings.

For Twitter and ACE 2008, we report the standard B^3 metric (Bagga and Baldwin, 1998a). For the political blog dataset, the reference does not consist of entity annotations, and so we follow the evaluation procedure of Yogatama et al. (2012).

8.1 Twitter

Data. We use a novel corpus of Twitter posts discussing the 2013 Grammy Award ceremony. This is a challenging corpus, featuring many instances

[9] In our experiments, we run the clustering algorithm five times, initialized from samples chosen at random from the last 10% of the sampler run, and keep the clustering that achieved highest expected Rand score.

of name variation. The dataset consists of five splits (by entity), the smallest of which is 604 mentions and the largest is 1374. We reserve the largest split for development purposes, and report our results on the remaining four. Appendix B provides more detail about the dataset.

Baselines. We use the discriminative entity clustering algorithm of Green et al. (2012) as our baseline; their approach was found to outperform another generative model which produced a flat clustering of mentions via a Dirichlet process mixture model. Their method uses Jaro-Winkler string similarity to match names, then clusters mentions with matching names (for disambiguation) by comparing their unigram context distributions using the Jenson-Shannon metric. We also compare to the EXACT-MATCH baseline, which assigns all strings with the same name to the same entity.

Procedure. We run four test experiments in which one split is used to pick model hyperparameters and the remaining three are used for test. For the discriminative baseline, we tune the string match threshold, context threshold, and the weight of the context model prior (all via grid search). For our model, we tune *only* the fixed weight of the root feature, which determines the precision/recall tradeoff (larger values of this feature result in more attachments to $\diamondsuit$ and hence more entities). We leave other hyperparameters fixed: 16 latent topics, and Gaussian priors $\mathcal{N}(0, 1)$ on all log-linear parameters. For PHYLO, the entity clustering is the result of (1) training the model using EM, (2) sampling from the posterior to obtain a distribution over clusterings, and (3) finding a consensus clustering. We use 20 iterations of EM with 100 samples per E-step for training, and use 1000 samples after training to estimate the posterior. We report results using three variations of our model: PHYLO does not consider mention context (all mentions effectively have the same topic) and determines mention entities from a single sample of $\boldsymbol{p}$ (the last); PHYLO+TOPIC adds context (§5.2); PHYLO+TOPIC+MBR uses the full posterior and consensus clustering to pick the output clustering (§7). Our results are shown in Table 1.[10]

[10]Our single-threaded implementation took around 15 minutes per fold of the Twitter corpus on a personal laptop with a 2.3 Ghz Intel Core i7 processor (including time required to parse the data files). Typical acceptance rates for ordering and topic proposals ranged from 0.03 to 0.08.

	Mean Test B^3		
	P	R	F1
EXACT-MATCH	99.6	53.7	69.8
Green et al. (2012)	92.1	69.8	79.3
PHYLO	85.3	91.4	88.7
PHYLO+TOPIC	92.8	90.8	91.8
PHYLO+TOPIC+MBR	92.9	90.9	91.9

Table 1: Results for the Twitter dataset. Higher B^3 scores are better. Note that each number is averaged over four different test splits. In three out of four experiments, PHYLO+TOPIC+MBR achieved the highest F1 score; in one case PHYLO+TOPIC won by a small margin.

		Test B^3		
		P	R	F1
PER	EXACT-MATCH	98.0	81.2	88.8
	Green et al. (2012)	95.0	88.9	91.9
	PHYLO+TOPIC+MBR	97.2	88.6	92.7
ORG	EXACT-MATCH	98.2	78.3	87.1
	Green et al. (2012)	92.1	88.5	90.3
	PHYLO+TOPIC+MBR	95.5	80.9	87.6

Table 2: Results for the ACE 2008 newswire dataset.

8.2 Newswire

Data. We use the ACE 2008 dataset, which is described in detail in Green et al. (2012). It is split into a development portion and a test portion. The baseline system took the first mention from each (gold) within-document coreference chain as the canonical mention, ignoring other mentions in the chain; we follow the same procedure in our experiments.[11]

Baselines & Procedure. We use the same baselines as in §8.1. On development data, modeling pragmatics as in §4.2 gave large improvements for organizations (8 points in F-measure), correcting the tendency to assume that short names like CIA were coincidental homonyms. Hence we allowed $\gamma > 0$ and tuned it on development data.[12] Results are in Table 2.

8.3 Blogs

Data. The CMU political blogs dataset consists of 3000 documents about U.S. politics (Yano et al., 2009). Preprocessed as described in Yogatama et al. (2012), the data consists of 10647 entity mentions.

[11]That is, each within-document coreference chain is mapped to a single mention as a preprocessing step.

[12]We used only a simplified version of the pragmatic model, approximating $w(p', x)$ as 1 or 0 according to whether $p'.n = x.n$. We also omitted the IMH step from section 5.3. The other results we report do not use pragmatics at all, since we found that it gave only a slight improvement on Twitter.

Unlike our other datasets, mentions are not annotated with entities: the reference consists of a table of 126 entities, where each row is the *canonical name* of one entity.

Baselines. We compare to the system results reported in Figure 2 of Yogatama et al. (2012). This includes a baseline hierarchical clustering approach, the "EEA" name canonicalization system of Eisenstein et al. (2011), as well the model proposed by Yogatama et al. (2012). Like the output of our model, the output of their hierarchical clustering baseline is a mention clustering, and therefore must be mapped to a table of canonical entity names to compare to the reference table.

Procedure & Results We tune our method as in previous experiments, on the initialization data used by Yogatama et al. (2012) which consists of a subset of 700 documents of the full dataset. The tuned model then produced a mention clustering on the full political blog corpus. As the mapping from clusters to a table is not fully detailed in Yogatama et al. (2012), we used a simple heuristic: the *most frequent* name in each cluster is taken as the canonical name, augmented by any titles from a predefined list appearing in any other name in the cluster. The resulting table is then evaluated against the reference, as described in Yogatama et al. (2012). We achieved a response score of 0.17 and a reference score of 0.61. Though not state-of-the-art, this result is close to the score of the "EEA" system of Eisenstein et al. (2011), as reported in Figure 2 of Yogatama et al. (2012), which is specifically designed for the task of canonicalization.

8.4 Discussion

On the Twitter dataset, we obtained a 12.6-point F1 improvement over the baseline. To understand our model's behavior, we looked at the sampled phylogenetic trees on development data. One reason our model does well in this noisy domain is that it is able to relate seemingly dissimilar names via successive steps. For instance, our model learned to relate many variations of LL Cool J:

Cool James	LLCoJ	El-El Cool John
LL	LL COOL JAMES	LLCOOLJ

In the sample we inspected, these mentions were also assigned the same topic, further boosting the probability of the configuration.

The ACE dataset, consisting of editorialized newswire, naturally contains less name variation than Twitter data. Nonetheless, we find that the variation that does appear is often properly handled by our model. For instance, we see several instances of variation due to transliteration that were all correctly grouped together, such as Megawati Soekarnoputri and Megawati Sukarnoputri. The pragmatic model was also effective in grouping common acronyms into the same entity.

We found that multiple samples tend to give different phylogenies (so the sampler is mobile), but essentially the same clustering into entities (which is why consensus clustering did not improve much over simply using the last sample). Random restarts of EM might create more variety by choosing different locally optimal parameter settings. It may also be beneficial to explore other sampling techniques (Bouchard-Côté, 2014).

Our method assembles observed names into an evolutionary tree. However, the true tree must include many names that fall outside our small observed corpora, so our model would be a more appropriate fit for a far larger corpus. Larger corpora also offer stronger signals that might enable our Monte Carlo methods to mix faster and detect regularities more accurately.

A common error of our system is to connect mentions that share long substrings, such as different PERSONs who share a last name, or different ORGANIZATIONs that contain University of. A more powerful name mutation than the one we use here would recognize entire words, for example inserting a common title or replacing a first name with its common nickname. Modeling the *internal structure* of names (Johnson, 2010; Eisenstein et al., 2011; Yogatama et al., 2012) in the mutation model is a promising future direction.

9 Conclusions

Our primary contribution consists of new modeling ideas, and associated inference techniques, for the problem of cross-document coreference resolution. We have described how writers systematically plunder (ϕ) and then systematically modify (θ) the work of past writers. Inference under such models could also play a role in tracking evolving memes and social influence, not merely in establishing strict coreference. Our model also provides an alternative to the distance-dependent CRP.[2]

Our implementation is available for research use at: `https://bitbucket.org/noandrews/phyloinf`.

References

Nicholas Andrews, Jason Eisner, and Mark Dredze. 2012. Name phylogeny: A generative model of string variation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 344–355, Jeju, Korea, July.

Amit Bagga and Breck Baldwin. 1998a. Algorithms for scoring coreference chains. In *In The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*, pages 563–566.

Amit Bagga and Breck Baldwin. 1998b. Entity-based cross-document coreferencing using the vector space model. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, ACL '98, pages 79–85, Stroudsburg, PA, USA. Association for Computational Linguistics.

Alex Baron and Marjorie Freedman. 2008. Who is who and what is what: Experiments in cross-document co-reference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 274–283, Stroudsburg, PA, USA. Association for Computational Linguistics.

Peter J. Bickel and Kjell A. Doksum. 1977. *Mathematical Statistics : Basic Ideas and Selected Topics*. Holden-Day, Inc.

David M. Blei and Peter I. Frazier. 2011. Distance dependent chinese restaurant processes. *J. Mach. Learn. Res.*, 12:2461–2488, November.

D. M. Blei, A. Y. Ng, and M. I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Alexandre Bouchard-Côté, David Hall, Thomas L. Griffiths, and Dan Klein. 2013. Automated reconstruction of ancient languages using probabilistic models of sound change. *Proceedings of the National Academy of Sciences*.

Alexandre Bouchard-Côté. 2014. Sequential Monte Carlo (SMC) for Bayesian phylogenetics. *Bayesian phylogenetics: methods, algorithms, and applications*.

William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. 2003. A comparison of string metrics for matching names and records. In *KDD Workshop on data cleaning and object consolidation*.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

C.H.Q. Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and H.D. Simon. 2001. A min-max cut algorithm for graph partitioning and data clustering. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 107 –114.

Mark Dredze, Michael J Paul, Shane Bergsma, and Hieu Tran. 2013. Carmen: A twitter geolocation system with applications to public health. In *AAAI Workshop on Expanding the Boundaries of Health Informatics Using AI (HIAI)*.

Markus Dreyer, Jason Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1080–1089, Honolulu, Hawaii, October. Association for Computational Linguistics.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1971–1982. Association for Computational Linguistics.

Jacob Eisenstein, Tae Yano, William W. Cohen, Noah A. Smith, and Eric P. Xing. 2011. Structured databases of named entities from bayesian nonparametrics. In *Proceedings of the First Workshop on Unsupervised Learning in NLP*, EMNLP '11, pages 2–12, Stroudsburg, PA, USA. Association for Computational Linguistics.

T. Finin, Z. Syed, J. Mayfield, P. McNamee, and C. Piatko. 2009. Using Wikitology for cross-document entity coreference resolution. In *AAAI Spring Symposium on Learning by Reading and Learning to Read*.

Spence Green, Nicholas Andrews, Matthew R. Gormley, Mark Dredze, and Christopher D. Manning. 2012. Entity clustering across languages. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 60–69, Stroudsburg, PA, USA. Association for Computational Linguistics.

Thomas L Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235.

Stephen Guo, Ming-Wei Chang, and Emre Kıcıman. 2013. To link or not to link? a study on end-to-end tweet entity linking. In *Proceedings of NAACL-HLT*, pages 1020–1030.

Aria Haghighi and Dan Klein. 2010. Coreference resolution in a modular, entity-centered model. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 385–393, Los Angeles, California, June. Association for Computational Linguistics.

Mark Johnson. 2010. Pcfgs, topic models, adaptor grammars and learning topical collocations and the structure of proper names. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1148–1157, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zornitsa Kozareva and Sujith Ravi. 2011. Unsupervised name ambiguity resolution using a generative model. In *Proceedings of the First Workshop on Unsupervised Learning in NLP*, EMNLP '11, pages 105–112, Stroudsburg, PA, USA. Association for Computational Linguistics.

Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky. 2012. Joint entity and event coreference resolution across documents. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Richard A. Levine and George Casella. 2001. Implementations of the Monte Carlo EM Algorithm. *Journal of Computational and Graphical Statistics*, 10(3):422–439.

Feiping Nie, Chris H. Q. Ding, Dijun Luo, and Heng Huang. 2010. Improved minmax cut graph clustering with nonnegative relaxation. In José L. Balcázar, Francesco Bonchi, Aristides Gionis, and Michèle Sebag, editors, *ECML/PKDD (2)*, volume 6322 of *Lecture Notes in Computer Science*, pages 451–466. Springer.

E. H. Porter and W. E. Winkler, 1997. *Approximate String Comparison and its Effect on an Advanced Record Linkage System*, chapter 6, pages 190–199. U.S. Bureau of the Census.

William M. Rand. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850.

Delip Rao, Paul McNamee, and Mark Dredze. 2010. Streaming cross document entity coreference resolution. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 1050–1058, Stroudsburg, PA, USA. Association for Computational Linguistics.

Eric Sven Ristad and Peter N. Yianilos. 1996. Learning string edit distance. Technical Report CS-TR-532-96, Princeton University, Department of Computer Science.

Eric Sven Ristad and Peter N. Yianilos. 1998. Learning string edit distance. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 20(5):522–532, May.

Alan Ritter, Sam Clark, Oren Etzioni, et al. 2011. Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534. Association for Computational Linguistics.

Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. 2011. Large-scale cross-document coreference using distributed inference and hierarchical models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 793–803, Portland, Oregon, USA, June. Association for Computational Linguistics.

R E Tarjan. 1977. Finding optimum branchings. *Networks*, 7(1):25–35.

Luke Tierney. 1994. Markov Chains for Exploring Posterior Distributions. *The Annals of Statistics*, 22(4):1701–1728.

Hanna Wallach, David Mimno, and Andrew McCallum. 2009. Rethinking lda: Why priors matter. In *Advances in Neural Information Processing Systems*, pages 1973–1981.

Michael Wick, Sameer Singh, and Andrew McCallum. 2012. A discriminative hierarchical model for fast coreference at large scale. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 379–388, Stroudsburg, PA, USA. Association for Computational Linguistics.

William E. Winkler. 1999. The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Census Bureau.

Tae Yano, William W. Cohen, and Noah A. Smith. 2009. Predicting response to political blog posts with topic models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 477–485, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dani Yogatama, Yanchuan Sim, and Noah A. Smith. 2012. A probabilistic model for canonicalizing named entity mentions. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 685–693, Stroudsburg, PA, USA. Association for Computational Linguistics.

Linguistic Structured Sparsity in Text Categorization

Dani Yogatama
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
dyogatama@cs.cmu.edu

Noah A. Smith
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
nasmith@cs.cmu.edu

Abstract

We introduce three linguistically motivated structured regularizers based on parse trees, topics, and hierarchical word clusters for text categorization. These regularizers impose linguistic bias in feature weights, enabling us to incorporate prior knowledge into conventional bag-of-words models. We show that our structured regularizers consistently improve classification accuracies compared to standard regularizers that penalize features in isolation (such as lasso, ridge, and elastic net regularizers) on a range of datasets for various text prediction problems: topic classification, sentiment analysis, and forecasting.

1 Introduction

What is the best way to exploit linguistic information in statistical text processing models? For tasks like text classification, sentiment analysis, and text-driven forecasting, this is an open question, as cheap "bag-of-words" models often perform well. Much recent work in NLP has focused on linguistic feature engineering (Joshi et al., 2010) or representation learning (Glorot et al., 2011; Socher et al., 2013).

In this paper, we propose a radical alternative. We embrace the conventional bag-of-words representation of text, instead bringing linguistic bias to bear on *regularization*. Since the seminal work of Chen and Rosenfeld (2000), the importance of regularization in discriminative models of text—including language modeling, structured prediction, and classification—has been widely recognized. The emphasis, however, has largely been on one specific kind of inductive bias: avoiding large weights (i.e., coefficients in a linear model).

Recently, *structured* (or composite) regularization has been introduced; simply put, it reasons about different weights jointly. The most widely explored variant, group lasso (Yuan and Lin, 2006) seeks to avoid large ℓ_2 norms for *groups* of weights. Group lasso has been shown useful in a range of applications, including computational biology (Kim and Xing, 2008), signal processing (Lv et al., 2011), and NLP (Eisenstein et al., 2011; Martins et al., 2011; Nelakanti et al., 2013). For text categorization problems, Yogatama and Smith (2014) proposed groups based on sentences, an idea generalized here to take advantage of richer linguistic information.

In this paper, we show how linguistic information of various kinds—parse trees, thematic topics, and hierarchical word clusterings—can be used to construct group lasso variants that impose linguistic bias without introducing any new features. Our experiments demonstrate that structured regularizers can squeeze higher performance out of conventional bag-of-words models on seven out of eight of text categorization tasks tested, in six cases with more compact models than the best-performing unstructured-regularized model.

2 Notation

We represent each document as a feature vector $\mathbf{x} \in \mathbb{R}^V$, where V is the vocabulary size. x_v is the frequency of the vth word (i.e., this is a "bag of words" model).

Consider a linear model that predicts a binary response $y \in \{-1, +1\}$ given $\mathbf{x}$ and weight vector $\mathbf{w} \in \mathbb{R}^V$. We denote our training data of D documents in the corpus by $\{\mathbf{x}_d, y_d\}_{d=1}^D$. The goal of the learning procedure is to estimate $\mathbf{w}$ by minimizing the regularized training data loss:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \Omega(\mathbf{w}) + \textstyle\sum_{d=1}^D \mathcal{L}(\mathbf{x}_d, \mathbf{w}, y_d),$$

where $\mathcal{L}(\mathbf{x}, \mathbf{w}, y)$ is the loss function for document d and $\Omega(\mathbf{w})$ is the regularizer.

In this work, we use the log loss:

$$\mathcal{L}(\mathbf{x}_d, \mathbf{w}, y_d) = -\log(1 + \exp(-y_d \mathbf{w}^\top \mathbf{x}_d)),$$

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 786–796,
Baltimore, Maryland, USA, June 23-25 2014.

Other loss functions (e.g., hinge loss, squared loss) can also be used with any of the regularizers discussed in this paper.

Our focus is on the regularizer, $\Omega(\mathbf{w})$. For high dimensional data such as text, regularization is crucial to avoid overfitting.[1]

The usual starting points for regularization are the "lasso" (Tibshirani, 1996) and the "ridge" (Hoerl and Kennard, 1970), based respectively on the ℓ_1 and squared ℓ_2 norms:

$$\Omega_{las}(\mathbf{w}) = \lambda_{las}\|\mathbf{w}\|_1 = \lambda \sum_j |w_j|$$
$$\Omega_{rid}(\mathbf{w}) = \lambda_{rid}\|\mathbf{w}\|_2^2 = \lambda \sum_j w_j^2$$

Both methods disprefer weights of large magnitude; smaller (relative) magnitude means a feature (here, a word) has a smaller effect on the prediction, and zero means a feature has no effect.[2] The hyperparameter λ in each case is typically tuned on a development dataset. A linear combination of ridge and lasso is known as the elastic net (Zou and Hastie, 2005). The lasso, ridge, and elastic net are three strong baselines in our experiments.

3 Group Lasso

Structured regularizers penalize estimates of $\mathbf{w}$ in which collections of weights are penalized jointly. For example, in the group lasso (Yuan and Lin, 2006), predefined groups of weights (subvectors of $\mathbf{w}$) are encouraged to either go to zero (as a group) or not (as a group)—this is known as "group sparsity."[3]

The variant of group lasso we explore here uses an $\ell_{1,2}$ norm. Let g index the G predefined groups of weights and $\mathbf{w}_g$ denote the subvector of $\mathbf{w}$ containing weights for group g:

$$\Omega_{glas}(\mathbf{w}) = \lambda_{glas} \sum_{g=1}^{G} \lambda_g \|\mathbf{w}_g\|_2,$$

where λ_{glas} is a hyperparameter tuned on a development data, and λ_g is a group specific weight. Typically the groups are non-overlapping, which offers computational advantages, but this need not be the case (Jacob et al., 2009; Jenatton et al., 2011).

4 Structured Regularizers for Text

Past work applying the group lasso to NLP problems has considered four ways of defining the groups. Eisenstein et al. (2011) defined groups of coefficients corresponding to the same independent variable applied to different (continuous) output variables in multi-output regression. Martins et al. (2011) defined groups based on feature templates used in chunking and parsing tasks. Nelakanti et al. (2013) defined groups based on n-gram histories for language modeling. In each of these cases, the groups were defined based on information from feature *types* alone; given the features to be used, the groups were known.

Here we build on a fourth approach that exploits structure in the data.[4] Yogatama and Smith (2014) introduced the *sentence regularizer*, which uses patterns of word cooccurrence in the training data to define groups. We review this method, then apply the idea to three more linguistically informed structure in text data.

4.1 Sentence Regularizer

The sentence regularizer exploits sentence boundaries in each training document. The idea is to define a group $g_{d,s}$ for every sentence s in every training document d. The group contains coefficients for words that occur in its sentence. This means that a word is a member of one group for every distinct (training) sentence it occurs in, and that the regularizer is based on word tokens, not types as in the approach of Martins et al. (2011) and Nelakanti et al. (2013). The regularizer is:

$$\Omega_{sen}(\mathbf{w}) = \sum_{d=1}^{D} \sum_{s=1}^{S_d} \lambda_{d,s} \|\mathbf{w}_{d,s}\|_2,$$

where S_d is the number of sentences in document d. This regularizer results in tens of thousands to millions of heavily *overlapping* groups, since a standard corpus typically contains thousands to millions of sentences and many words that appear in more than one sentence.

[1] A Bayesian interpretation of regularization is as a prior on the weight vector $\mathbf{w}$; in many cases Ω can be understood as a log-prior representing beliefs about the model held before exposure to data. For lasso regression, the prior is a zero-mean Laplace distribution, whereas for ridge regression the prior is a zero-mean Gaussian distribution. For non-overlapping group lasso, the prior is a two-level hierarchical Bayes model (Figueiredo, 2002). The Bayesian interpretation of overlapping group lasso is not yet well understood.

[2] The lasso leads to strongly sparse solutions, in which many elements of the estimated $\mathbf{w}$ are actually zero. This is an attractive property for efficiency and (perhaps) interpretability. The ridge encourages weights to go toward zero, but usually not all the way *to* zero; for this reason its solutions are known as "weakly" sparse.

[3] Other structured regularizers include the fused lasso (Tibshirani et al., 2005) and the elitist lasso (Kowalski and Torresani, 2009).

[4] This provides a compelling reason *not* to view such methods in a Bayesian framework: if the regularizer is informed by the data, then it does not truly correspond to a prior.

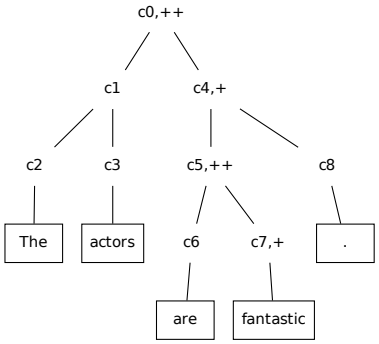


Figure 1: An example of a parse tree from the Stanford sentiment treebank, which annotates sentiment at the level of every constituent (indicated here by + and ++; no marking indicates neutral sentiment). The sentence is The actors are fantastic. Our regularizer constructs nine groups for this sentence, corresponding to $c_0, c_1, \ldots, c_8$. g_{c_0} consists of 5 weights—$\langle w_{the}, w_{actors}, w_{are}, w_{fantastic}, w_. \rangle$, exactly the same as the group in the sentence regularizer—g_{c_1} consists of 2 words, g_{c_4} of 3 words, etc. Notice that c_2, c_3, c_6, c_7, and c_8 each consist of only 1 word. The Stanford sentiment treebank has an annotation of sentiments at the constituent level. As in this example, most constituents are annotated as neutral.

If the norm of $\mathbf{w}_{g_{d,s}}$ is driven to zero, then the learner has deemed the corresponding sentence irrelevant to the prediction. It is important to point out that, while the regularizer prefers to zero out the weights for all words in irrelevant sentences, it also prefers *not* to zero out weights for words in relevant sentences. Since the groups overlap and may work against each other, the regularizer may not be able to drive many weights to zero on its own. Yogatama and Smith (2014) used a linear combination of the sentence regularizer and the lasso (a kind of *sparse group lasso*; Friedman et al., 2010) to also encourage weights of irrelevant word types to go to zero.[5]

4.2 Parse Tree Regularizer

Sentence boundaries are a rather superficial kind of linguistic structure; syntactic parse trees provide more fine-grained information. We introduce a new regularizer, the parse tree regularizer, in which groups are defined for every constituent in every parse of a training data sentence.

Figure 1 illustrates the group structures derived from an example sentence from the Stanford sentiment treebank (Socher et al., 2013). This regularizer captures the idea that *phrases* might be selected as relevant or (in most cases) irrelevant to a task, and is expected to be especially useful in sentence-level prediction tasks.

The parse-tree regularizer (omitting the group coefficients and λ) for one sentence with the parse tree shown in Figure 1 is:

$$\Omega_{tree}(\mathbf{w}) = \sqrt{|w_{the}|^2 + |w_{actors}|^2 + |w_{are}|^2 + |w_{fantastic}|^2 + |w_.|^2} + \sqrt{|w_{are}|^2 + |w_{fantastic}|^2 + |w_.^2|} + \sqrt{|w_{the}|^2 + |w_{actors}|^2} + \sqrt{|w_{are}|^2 + |w_{fantastic}|^2} + |w_{the}| + |w_{actors}| + |w_{are}| + |w_{fantastic}| + |w_.|$$

The groups have a tree structure, in that assigning zero values to the weights in a group corresponding to a higher-level constituent implies the same for those constituents that are dominated by it. This resembles the tree-guided group lasso in Kim and Xing (2008), although the leaf nodes in their tree represent tasks in multi-task regression.

Of course, in a corpus there are many parse trees (one per sentence, so the number of parse trees is the number of sentences). The parse-tree regularizer is:

$$\Omega_{tree}(\mathbf{w}) = \sum_{d=1}^{D} \sum_{s=1}^{S_d} \sum_{c=1}^{C_{d,s}} \lambda_{d,s,c} \|\mathbf{w}_{d,s,c}\|_2,$$

where $\lambda_{d,s,c} = \lambda_{glas} \times \sqrt{size(g_{d,s,c})}$, d ranges over (training) documents and c ranges over constituents in the parse of sentence s in document d. Similar to the sentence regularizer, the parse-tree regularizer operates on word tokens. Note that, since each word token is itself a constituent, the parse tree regularizer includes terms just like the lasso naturally, penalizing the absolute value of each word's weight in isolation. For the lasso-like penalty on each word, instead of defining the group weights to be $1 \times$ the number of tokens for each word type, we tune one group weight for all word types on a development data. As a result, besides λ_{glas}, we have an additional hyperparameter, denoted by λ_{las}.

To gain an intuition for this regularizer, consider the case where we apply the penalty only for a single tree (sentence), which for ease of exposition is assumed not to use the same word more than once (i.e., $\|\mathbf{x}\|_\infty = 1$). Because it instantiates the tree-structured group lasso, the regularizer will require bigger constituents to be "included" (i.e., their words given nonzero weight) before smaller constituents can be included. The result is that some words may not be included. Of course, in some sentences, some words will occur more than once, and the parse tree regularizer instantiates groups for constituents in every sentence in the training corpus, and these groups may work against each other. The parse tree regularizer should therefore

[5] Formally, this is equivalent to including one additional group for each word type.

be understood as encouraging group behavior of syntactically grouped words, or sharing of information by syntactic neighbors.

In sentence level prediction tasks, such as sentence-level sentiment analysis, it is known that most constituents (especially those that correspond to shorter phrases) in a parse tree are uninformative (neutral sentiment). This was verified by Socher et al. (2013) when annotating phrases in a sentence for building the Stanford sentiment treebank. Our regularizer incorporates our prior expectation that most constituents should have no effect on prediction.

4.3 LDA Regularizer

Another type of structure to consider is topics. For example, if we want to predict whether a paper will be cited or not (Yogatama et al., 2011), the model can perform better if it knows beforehand the collections of words that represent certain themes (e.g., in ACL papers, these might include machine translation, parsing, etc.). As a result, the model can focus on which topics will increase the probability of getting citations, and penalize weights for words in the same topic together, instead of treating each word separately.

We do this by inferring topics in the training corpus by estimating the latent Dirichlet allocation (LDA) model (Blei et al., 2003)). Note that LDA is an unsupervised method, so we can infer topical structures from *any* collection of documents that are considered related to the target corpus (e.g., training documents, text from the web, etc.). This contrasts with typical semi-supervised learning methods for text categorization that combine unlabeled and labeled data within a generative model, such as multinomial naïve Bayes, via expectation-maximization (Nigam et al., 2000) or semi-supervised frequency estimation (Su et al., 2011). Our method does not use unlabeled data to obtain more training documents or estimate the joint distributions of words better, but it allows the use of unlabeled data to induce topics. We leave comparison with other semi-supervised methods for future work.

There are many ways to associate inferred topics with group structure. In our experiments, we choose the R most probable words given a topic and create a group for them.[6] The LDA regularizer can be written as:

$$\Omega_{lda}(\mathbf{w}) = \sum_{k=1}^{K} \lambda_k \|\mathbf{w}_k\|_2,$$

where k ranges over the K topics. Similar to our earlier notations, $\mathbf{w}_k$ corresponds to the subvector of $\mathbf{w}$ such that the corresponding features are present in topic k. Note that in this case we can also have overlapping groups, since words can appear in the top R of many topics.

$k=1$	$k=2$	$k=3$	$k=4$
soccer	injury	physics	monday
striker	knee	gravity	tuesday
midfielder	ligament	moon	april
goal	shoulder	sun	june
defender	cruciate	relativity	sunday

Table 1: A toy example of $K = 4$ topics. The top $R = 5$ words in each topics are displayed. The LDA regularizer will construct four groups from these topics. The first group is $\langle w_{soccer}, w_{striker}, w_{midfielder}, w_{goal}, w_{defender} \rangle$, the second group is $\langle w_{injury}, w_{knee}, w_{ligament}, w_{shoulder}, w_{cruciate} \rangle$, etc. In this example, there are no words occurring in the top R of more than one topic, but that need not be the case in general.

To gain an intuition for this regularizer, consider the toy example in Table 1. the case where we have $K = 4$ topics and we select $R = 5$ top words from each topic. Supposed that we want to classify whether an article is a sports article or a science article. The regularizer might encourage the weights for the fourth topics' words toward zero, since they are less useful for the task. Additionally, the regularizer will penalize words in each of the other three groups collectively. Therefore, if (for example) *ligament* is deemed a useful feature for classifying an article to be about sports, then the other words in that topic will have a smaller effective penalty for getting nonzero weights—even weights of the opposite sign as $w_{ligament}$. It is important to distinguish this from unstructured regularizers such as the lasso, which penalize each word's weight on its own without regard for related word types.

Unlike the parse tree regularizer, the LDA regularizer is not tree structured. Since the lasso-like penalty does not occur naturally in a non tree-structured regularizer, we add an additional lasso penalty for each word type (with hyperparameter λ_{las}) to also encourage weights of irrelevant words to go to zero. Our LDA regularizer is an instance of sparse group lasso (Friedman et al., 2010).

[6] Another possibility is to group the smallest set of words whose total probability given a topic amounts to P (e.g., 0.99). mass of a topic. Preliminary experiments found this not to work well.

4.4 Brown Cluster Regularizer

Brown clustering is a commonly used unsupervised method for grouping words into a hierarchy of clusters (Brown et al., 1992). Because it uses local information, it tends to discover words with similar syntactic behavior, though semantic groupings are often evident, especially at the more fine-grained end of the hierarchy.

We incorporate Brown clusters into a regularizer in a similar way to the topical word groups inferred using LDA in §4.3, but here we make use of the hierarchy. Specifically, we construct tree-structured groups, one per cluster (i.e., one per node in the hierarchy). The Brown cluster regularizer is:

$$\Omega_{brown}(\mathbf{w}) = \sum_{v=1}^{N} \lambda_v \|\mathbf{w}_v\|_2,$$

where v ranges over the N nodes in the Brown cluster tree. As a tree structured regularizer, this regularizer enforces constraints that a node v's group is given nonzero weights only if those nodes that dominate v (i.e., are on a path from v to the root) have their groups selected.

Consider a similar toy example to the LDA regularizer (sports vs. science) and the hierarchical clustering of words in Figure 2. In this case, the Brown cluster regularizer will create 17 groups, one for every node in the clustering tree. The regularizer for this tree (omitting the group coefficients and λ) is:

$$\begin{aligned}\Omega_{brown}(\mathbf{w}) = \sum_{i=0}^{7} \|\mathbf{w}_{v_i}\|_2 + |w_{goal}| + |w_{striker}| \\ + |w_{midfielder}| + |w_{knee}| + |w_{injury}| \\ + |w_{gravity}| + |w_{moon}| + |w_{sun}|\end{aligned}$$

The regularizer penalizes words in a cluster together, exploiting discovered syntactic relatedness. Additionally, the regularizer can zero out weights of words corresponding to any of the internal nodes, such as v_7 if the words `monday` and `sunday` are deemed irrelevant to prediction.

Note that the regularizer already includes terms like the lasso naturally. Similar to the parse tree regularizer, for the lasso-like penalty on each word, we tune one group weight for all word types on a development data with a hyperparameter λ_{las}.

A key difference between the Brown cluster regularizer and the parse tree regularizer is that there is only one tree for Brown cluster regularizer, whereas the parse tree regularizer can have millions (one per sentence in the training data). The

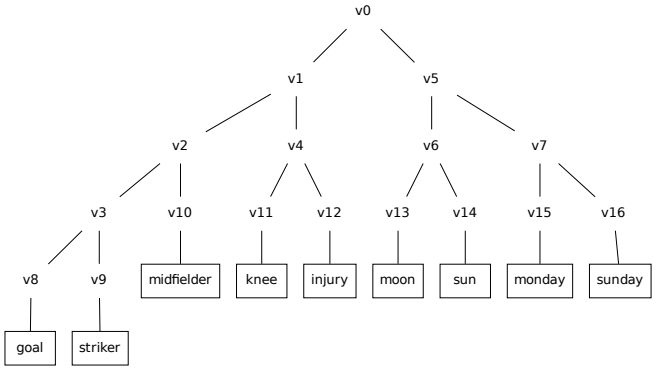


Figure 2: An illustrative example of Brown clusters for $N = 9$. The Brown cluster regularizer constructs 17 groups, one per node in for this tree, $v_0, v_1, \ldots, v_{16}$. v_0 contains 8 words, v_1 contains 5, etc. Note that the leaves, $v_8, v_9, \ldots, v_{16}$, each contain one word.

LDA and Brown cluster regularizers offer ways to incorporate unlabeled data, if we believe that the unlabeled data can help us infer better topics or clusters. Note that the processes of learning topics or clusters, or parsing training data sentences, are a separate stage that precedes learning our predictive model.

5 Learning

There are many optimization methods for learning models with structured regularizers, particulary group lasso (Jacob et al., 2009; Jenatton et al., 2011; Chen et al., 2011; Qin and Goldfarb, 2012; Yuan et al., 2013). We choose the optimization method of Yogatama and Smith (2014) since it handles millions of overlapping groups effectively. The method is based on the alternating directions method of multipliers (ADMM; Hestenes, 1969; Powell, 1969). We review it here in brief, for completeness, and show how it can be applied to tree-structured regularizers (such as the parse tree and Brown cluster regularizers in §4) in particular.

Our learning problem is, generically:

$$\min_{\mathbf{w}} \Omega(\mathbf{w}) + \sum_{d=1}^{D} \mathcal{L}(\mathbf{x}_d, \mathbf{w}, y_d).$$

Separating the lasso-like penalty for each word type from our group regularizers, we can rewrite this problem as:

$$\begin{aligned}\min_{\mathbf{w},\mathbf{v}} \Omega_{las}(\mathbf{w}) + \Omega_{glas}(\mathbf{v}) + \sum_{d=1}^{D} \mathcal{L}(\mathbf{x}_d, \mathbf{w}, y_d) \\ \text{s.t. } \mathbf{v} = \mathbf{M}\mathbf{w}\end{aligned}$$

where $\mathbf{v}$ consists of copies of the elements of $\mathbf{w}$. Notice that we work directly on $\mathbf{w}$ instead of the copies for the lasso-like penalty, since it does not have overlaps and has its own hyperparameters λ_{las}. For the remaining groups with size greater than one, we create copies $\mathbf{v}$ of size

$L = \sum_{g=1}^{G} \text{size}(g)$. $\mathbf{M} \in \{0,1\}^{L \times V}$ is a matrix whose 1s link elements of $\mathbf{w}$ to their copies.[7] We now have a constrained optimization problem, from which we can create an augmented Lagrangian problem; let $\mathbf{u}$ be the Lagrange variables:

$$\Omega_{las}(\mathbf{w}) + \Omega_{glas}(\mathbf{v}) + \mathcal{L}(\mathbf{w}) + \mathbf{u}^\top(\mathbf{v} - \mathbf{M}\mathbf{w}) + \frac{\rho}{2}\|\mathbf{v} - \mathbf{M}\mathbf{w}\|_2^2$$

ADMM proceeds by iteratively updating each of $\mathbf{w}$, $\mathbf{v}$, and $\mathbf{u}$, amounting to the following subproblems:

$$\min_{\mathbf{w}} \Omega_{las}(\mathbf{w}) + \mathcal{L}(\mathbf{w}) - \mathbf{u}^\top\mathbf{M}\mathbf{w} + \frac{\rho}{2}\|\mathbf{v} - \mathbf{M}\mathbf{w}\|_2^2 \quad (1)$$

$$\min_{\mathbf{v}} \Omega_{glas}(\mathbf{v}) + \mathbf{u}^\top\mathbf{v} + \frac{\rho}{2}\|\mathbf{v} - \mathbf{M}\mathbf{w}\|_2^2 \quad (2)$$

$$\mathbf{u} = \mathbf{u} + \rho(\mathbf{v} - \mathbf{M}\mathbf{w}) \quad (3)$$

Yogatama and Smith (2014) show that Eq. 1 can be rewritten in a form quite similar to ℓ_2-regularized loss minimization.[8]

Eq. 2 is the proximal operator of $\frac{1}{\rho}\Omega_{glas}$ applied to $\mathbf{M}\mathbf{w} - \frac{\mathbf{u}}{\rho}$. As such, it depends on the form of $\mathbf{M}$. Note that when applied to the collection of "copies" of the parameters, $\mathbf{v}$, Ω_{glas} no longer has overlapping groups. Defined $\mathbf{M}_g$ as the rows of $\mathbf{M}$ corresponding to weight copies assigned to group g. Let $\mathbf{z}_g \triangleq \mathbf{M}_g\mathbf{w} - \frac{\mathbf{u}_g}{\rho}$. Denote $\lambda_g = \lambda_{glas}\sqrt{\text{size}(g)}$. The problem can be solved by applying the proximal operator used in non-overlapping group lasso to each subvector:

$$\mathbf{v}_g = \text{prox}_{\Omega_{glas}, \frac{\lambda_g}{\rho}}(\mathbf{z}_g) = \begin{cases} \mathbf{0} & \text{if } \|\mathbf{z}_\mathbf{g}\|_2 \leq \frac{\lambda_g}{\rho} \\ \frac{\|\mathbf{z}_g\|_2 - \frac{\lambda_g}{\rho}}{\|\mathbf{z}_g\|_2}\mathbf{z}_g & \text{otherwise.} \end{cases}$$

For a tree structured regularizer, we can get speedups by working from the root node towards the leaf nodes when applying the proximal operator in the second step. If g is a node in a tree which is driven to zero, all of its children h that has $\lambda_h \leq \lambda_g$ will also be driven to zero.

Eq. 3 is a simple update of the dual variable $\mathbf{u}$. Algorithm 1 summarizes our learning procedure.[9]

[7] For the parse tree regularizer, L is the sum, over all training-data word tokens t, of the number of constituents t belongs to. For the LDA regularizer, $L = R \times K$. For the Brown cluster regularizer, $L = V - 1$.

[8] The difference lies in that the squared ℓ_2 norm in the penalty penalizes the difference between $\mathbf{w}$ and a vector that depends on the current values of $\mathbf{u}$ and $\mathbf{v}$. This does not affect the algorithm or its convergence in any substantive way.

[9] We use relative changes in the ℓ_2 norm of the parameter vector $\mathbf{w}$ as our convergence criterion (threshold of 10^{-3}), and set the maximum number of iterations to 100. Other criteria can also be used.

Algorithm 1 ADMM for overlapping group lasso

Input: augmented Lagrangian variable ρ, regularization strengths λ_{glas} and λ_{las}
while stopping criterion not met **do**
  $\mathbf{w} = \arg\min_{\mathbf{w}} \Omega_{las}(\mathbf{w}) + \mathcal{L}(\mathbf{w}) + \frac{\rho}{2}\sum_{i=1}^{V} N_i(w_i - \mu_i)^2$
  for $g = 1$ **to** G **do**
    $\mathbf{v}_g = \text{prox}_{\Omega_{glas}, \frac{\lambda_g}{\rho}}(\mathbf{z}_g)$
  end for
  $\mathbf{u} = \mathbf{u} + \rho(\mathbf{v} - \mathbf{M}\mathbf{w})$
end while

6 Experiments

6.1 Datasets

We use publicly available datasets to evaluate our model described in more detail below.

Topic classification. We consider four binary categorization tasks from the 20 Newsgroups dataset.[10] Each task involves categorizing a document according to two related categories: comp.sys: ibm.pc.hardware vs. mac.hardware; rec.sport: baseball vs. hockey; sci: med vs. space; and alt.atheism vs. soc.religion.christian.

Sentiment analysis. One task in sentiment analysis is predicting the polarity of a piece of text, i.e., whether the author is favorably inclined toward a (usually known) subject of discussion or proposition (Pang and Lee, 2008). Sentiment analysis, even at the coarse level of polarity we consider here, can be confused by negation, stylistic use of irony, and other linguistic phenomena. Our sentiment analysis datasets consist of movie reviews from the Stanford sentiment treebank (Socher et al., 2013),[11] and floor speeches by U.S. Congressmen alongside "yea"/"nay" votes on the bill under discussion (Thomas et al., 2006).[12] For the Stanford sentiment treebank, we only predict binary classifications (positive or negative) and exclude neutral reviews.

Text-driven forecasting. Forecasting from text requires identifying textual correlates of a response variable revealed in the future, most of which will be weak and many of which will be spurious (Kogan et al., 2009). We consider two such problems. The first one is predicting whether a scientific paper will be cited or not within three years of its publication (Yogatama et al., 2011);

[10] http://qwone.com/~jason/20Newsgroups

[11] http://nlp.stanford.edu/sentiment/

[12] http://www.cs.cornell.edu/~ainur/data.html

	Dataset	D	# Dev.	# Test	V
20N	science	952	235	790	30,154
	sports	958	239	796	20,832
	relig.	870	209	717	24,528
	comp.	929	239	777	20,868
Sent.	movie	6,920	872	1,821	17,576
	vote	1,175	257	860	24,508
Fore.	science	3,207	280	539	42,702
	bill	37,850	7,341	6,571	10,001

Table 2: Descriptive statistics about the datasets.

the dataset comes from the ACL Anthology and consists of research papers from the Association for Computational Linguistics and citation data (Radev et al., 2009). The second task is predicting whether a legislative bill will be recommended by a Congressional committee (Yano et al., 2012).[13]

Table 2 summarizes statistics about the datasets used in our experiments. In total, we evaluate our method on eight binary classification tasks.

6.2 Setup

In all our experiments, we use unigram features plus an additional bias term which is not regularized. We compare our new regularizers with state-of-the-art methods for document classification: lasso, ridge, and elastic net regularization, as well as the sentence regularizer discussed in §4.1 (Yogatama and Smith, 2014).[14]

We parsed all corpora using the Berkeley parser (Petrov and Klein, 2007).[15] For the LDA regularizers, we ran LDA[16] on training documents with $K = 1,000$ and $R = 10$. For the Brown cluster regularizers, we ran Brown clustering[17] on training documents with $5,000$ clusters for the topic classification and sentiment analysis datasets, and $1,000$ for the larger text forecasting datasets (since they are bigger datasets that took more time).

[13] http://www.ark.cs.cmu.edu/bills

[14] Hyperparameters are tuned on a separate development dataset, using accuracy as the evaluation criterion. For lasso and ridge models, we choose λ from $\{10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3\}$. For elastic net, we perform grid search on the same set of values as ridge and lasso experiments for λ_{rid} and λ_{las}. For the sentence, Brown cluster, and LDA regularizers, we perform grid search on the same set of values as ridge and lasso experiments for $\rho, \lambda_{glas}, \lambda_{las}$. For the parse tree regularizer, because there are many more groups than other regularizers, we choose λ_{glas} from $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10\}$, ρ and λ_{las} from the same set of values as ridge and lasso experiments. If there is a tie on development data we choose the model with the smallest number of nonzero weights.

[15] https://code.google.com/p/berkeleyparser/

[16] http://www.cs.princeton.edu/~blei/lda-c/

[17] https://github.com/percyliang/brown-cluster

6.3 Results

Table 3 shows the results of our experiments on the eight datasets. The results demonstrate the superiority of structured regularizers. One of them achieved the best result on all but one dataset.[18] It is also worth noting that in most cases all variants of the structured regularizers outperformed lasso, ridge, and elastic net. In four cases, the new regularizers in this paper outperform the sentence regularizer.

We can see that the parse tree regularizer performed the best for the movie review dataset. The task is to predict sentence-level sentiment, so each training example is a sentence. Since constituent-level annotations are available for this dataset, we only constructed groups for neutral constituents (i.e., we drive neutral constituents to zero during training). It has been shown that syntactic information is helpful for sentence-level predictions (Socher et al., 2013), so the parse tree regularizer is naturally suitable for this task.

The Brown cluster and LDA regularizers performed best for the forecasting scientific articles dataset. The task is to predict whether an article will be cited or not within three years after publication. Regularizers that exploit the knowledge of semantic relations (e.g., topical categories), such as the Brown cluster and LDA regularizers, are therefore suitable for this type of prediction.

Table 4 shows model sizes obtained by each of the regularizers for each dataset. While lasso prunes more aggressively, it almost always performs worse. Our structured regularizers were able to obtain a significantly smaller model (27%, 34%, 19% as large on average for parse tree, Brown, and LDA regularizers respectively) compared to the ridge model.

Topic and cluster features. Another way to incorporate LDA topics and Brown clusters into a linear model is by adding them as additional features. For the 20N datasets, we also ran lasso, ridge, and elastic net with *additional* LDA topic and Brown cluster features.[19] Note that these new baselines use more features than our model. We can also add these additional features to our model

[18] This "bill" dataset, where they offered no improvement, is the largest by far (37,850 documents), and therefore the one where regularizers should matter the least. Note that the differences are small across regularizers for this dataset.

[19] For LDA, we took the top 10 words in a topic as a feature. For Brown clusters, we add a cluster as an additional feature if its size is less than 50.

Task	Dataset	Accuracy (%)							
		m.f.c.	lasso	ridge	elastic	sentence	parse	Brown	LDA
20N	science	50.13	90.63	91.90	91.65	**96.20**	92.66	93.04	93.67
	sports	50.13	91.08	93.34	93.71	**95.10**	93.09	93.71	94.97
	religion	55.51	90.52	92.47	92.47	92.75	**94.98**	92.89	93.03
	computer	50.45	85.84	86.74	87.13	**90.86**	89.45	86.36	88.42
Sentiment	movie	50.08	78.03	80.45	80.40	80.72	**81.55**	80.34	78.36
	vote	58.37	73.14	72.79	72.79	**73.95**	73.72	66.86	73.14
Forecasting	science	50.28	64.00	66.79	66.23	67.71	66.42	69.02	**69.39**
	bill	87.40	88.36	87.70	**88.48**	88.11	87.98	88.20	88.27

Table 3: Classification accuracies on various datasets. "m.f.c." is the most frequent class baseline. Boldface shows best results.

Task	Dataset	Model size (%)							
		m.f.c.	lasso	ridge	elastic	sentence	parse	Brown	LDA
20N	science	-	1	100	34	12	2	42	9
	sports	-	2	100	15	3	3	16	9
	religion	-	0.3	100	48	94	72	41	15
	computer	-	2	100	24	10	5	24	8
Sentiment	movie	-	10	100	54	83	87	59	12
	vote	-	2	100	44	6	2	30	4
Forecasting	science	-	31	100	43	99	9	50	90
	bill	-	7	100	7	8	37	7	7

Table 4: Model sizes (percentages of nonzero features in the resulting models) on various datasets.

Dataset	+ LDA features			LDA reg.
	lasso	ridge	elastic	
science	90.63	91.90	91.90	**93.67**
sports	91.33	93.47	93.84	**94.97**
religion	91.35	92.47	91.35	**93.03**
computer	85.20	86.87	86.35	**88.42**

Dataset	+ Brown features			Brown reg.
	lasso	ridge	elastic	
science	86.96	90.51	91.14	**93.04**
sports	82.66	88.94	85.43	**93.71**
religion	94.98	**96.93**	**96.93**	92.89
computer	55.72	**96.65**	67.57	86.36

Table 5: Classification accuracies on the 20N datasets for lasso, ridge, and elastic net models with additional LDA features (top) and Brown cluster features (bottom). The last column shows structured regularized models from Table 3.

and treat them as regular features (i.e., they do not belong to any groups and are regularized with standard regularizer such as the lasso penalty). The results in Table 5 show that for these datasets, models that incorporate this information through structured regularizers outperformed models that encode this information as additional features in 4 out 4 of cases (LDA) and 2 out of 4 cases (Brown). Sparse models with Brown clusters appear to overfit badly; recall that the clusters were learned on only the training data—clusters from a larger dataset would likely give stronger results. Of course, better performance might also be achieved by incorporating new features as well as using structured regularizers.

6.4 Examples

To gain an insight into the models, we inspect group sparsity patterns in the learned models by looking at the parameter copies $\mathbf{v}$. This lets us see which groups are considered important (i.e., "selected" vs. "removed"). For each of the proposed regularizers, we inspect the model a task in which it performed well.

For the parse tree regularizer, we inspect the model for the 20N:religion task. We observed that the model included most of the sentences (root node groups), but in some cases removed phrases from the parse trees, such as *ozzy osbourne* in the sentence *ozzy osbourne , ex-singer and main character of the black sabbath of good ole days past , is and always was a devout catholic .*

For the LDA regularizer, we inspect zero and nonzero groups (topics) in the forecasting scientific articles task. In this task, we observed that 642 out of 1,000 topics are driven to zero by our model. Table 6 shows examples of zero and nonzero topics for the dev.-tuned hyperparameter values. We can see that in this particular case, the model kept meaningful topics such as parsing and speech processing, and discarded general topics that are not correlated with the content of the papers (e.g., acknowledgment, document metadata, equation, etc.). Note that most weights for non-selected groups, even in $\mathbf{w}$, are near zero.

For the Brown cluster regularizer, we inspect the model from the 20N:science task. 771 out of 5,775 groups were driven to zero for the best model tuned on the development set. Examples of zero and nonzero groups are shown in Table 7. Similar to the LDA example, the groups that were driven to zero tend to contain generic words that are not relevant to the predictions. We can also see the tree structure effect in the regularizer. The group {*underwater*, *industrial*} was

$= 0$	"acknowledgment": workshop arpa program session darpa research papers spoken technology systems "document metadata": university references proceedings abstract work introduction new been research both "equation": pr w h probability wi gram context z probabilities complete "translation": translation target source german english length alignment hypothesis translations position
$\neq 0$	"translation": **korean** translation english rules sentences parsing input evaluation machine verb "speech processing": speaker **identification** topic recognition recognizer models acoustic test vocabulary independent "parsing": parser parsing **probabilistic** prediction parse **pearl** edges chart phase theory "classification": documents learning accuracy **bayes classification** wt document **naive** method selection

Table 6: Examples of LDA regularizer-removed and -selected groups (in $\mathbf{v}$) in the forecasting scientific articles dataset. Words with weights (in $\mathbf{w}$) of magnitude greater than 10^{-3} are highlighted in **red** (not cited) and **blue** (cited).

$= 0$	underwater industrial spotted hit reaped rejuvenated destroyed stretched undertake shake run seeing developing tingles diminishing launching finding investigating receiving maintaining adds engage explains builds
$\neq 0$	failure reproductive **ignition** reproduction **cyanamid planetary nikola** fertility **astronomical** geophysical # **lunar** cometary **supplying** astronautical **magnetic** atmospheric **std** underwater **hpr** wordscan exclusively aneutronic industrial peoples **obsessive** congenital **rare simple bowel** hereditary breast

Table 7: Examples of Brown regularizer-removed and -selected groups (in $\mathbf{v}$) in the 20N:science task. # denotes any numeral. Words with weights (in $\mathbf{w}$) of magnitude greater than 10^{-3} are highlighted in **red** (space) and **blue** (medical).

driven to zero, but not once it combined with other words such as *hpr*, *std*, *obsessive*. Note that we ran Brown clustering only on the training documents; running it on a larger collection of (unlabeled) documents relevant to the prediction task (i.e., semi-supervised learning) is worth exploring in future work.

7 Related and Future Work

Overall, our results demonstrate that linguistic structure in the data can be used to improve bag-of-words models, through structured regularization. State-of-the-art approaches to some of these problems have used additional features and representations (Yessenalina et al., 2010; Socher et al., 2013). For example, for the vote sentiment analysis datasets, latent variable models of Yessenalina et al. (2010) achieved a superior result of 77.67%. To do so, they sacrificed convexity and had to rely on side information for initialization. Our experimental focus has been on a controlled comparison between regularizers for a fixed model family (the simplest available, linear with bag-of-words features). However, the improvements offered by our regularization methods can be applied in future work to other model families with more carefully engineered features, metadata features (especially important in forecasting), latent variables, etc. In particular, note that other kinds of weights (e.g., metadata) can be penalized conventionally, or incorporated into the structured regularization where it makes sense to do so (e.g., n-grams, as in Nelakanti et al., 2013).

8 Conclusion

We introduced three data-driven, linguistically informed structured regularizers based on parse trees, topics, and hierarchical word clusters. We empirically showed that models regularized using our methods consistently outperformed standard regularizers that penalize features in isolation such as lasso, ridge, and elastic net on a range of datasets for various text prediction problems: topic classification, sentiment analysis, and forecasting.

Acknowledgments

The authors thank Brendan O'Connor for help with visualization and three anonymous reviewers for helpful feedback on an earlier draft of this paper. This research was supported in part by computing resources provided by a grant from the Pittsburgh Supercomputing Center, a Google research award, and the Intelligence Advanced Research Projects Activity via Department of Interior National Business Center contract number D12PC00347. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

References

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479.

Stanley F. Chen and Ronald Rosenfeld. 2000. A survey of smoothing techniques for me models. *IEEE Transactions on Speech and Audio Processing*, 8(1):37–50.

Xi Chen, Qihang Lin, Seyoung Kim, Jaime G. Carbonell, and Eric P. Xing. 2011. Smoothing proximal gradient method for general structured sparse learning. In *Proc. of UAI*.

Jacob Eisenstein, Noah A. Smith, and Eric P. Xing. 2011. Discovering sociolinguistic associations with structured sparsity. In *Proc. of ACL*.

Mario A. T. Figueiredo. 2002. Adaptive sparseness using Jeffreys' prior. In *Proc. of NIPS*.

Jerome Friedman, Trevor Hastie, and Robert Tibshiran. 2010. A note on the group lasso and a sparse group lasso. Technical report, Stanford University.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proc. of ICML*.

Magnus R. Hestenes. 1969. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4:303–320.

Arthur E. Hoerl and Robert W. Kennard. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Laurent Jacob, Guillaume Obozinski, and Jean-Philippe Vert. 2009. Group lasso with overlap and graph lasso. In *Proc. of ICML*.

Rodolphe Jenatton, Jean-Yves Audibert, and Francis Bach. 2011. Structured variable selection with sparsity-inducing norms. *Journal of Machine Learning Research*, 12:2777–2824.

Mahesh Joshi, Dipanjan Das, Kevin Gimpel, and Noah A. Smith. 2010. Movie reviews and revenues: An experiment in text regression. In *Proc. of NAACL*.

Seyoung Kim and Eric P. Xing. 2008. Feature selection via block-regularized regression. In *Proc. of UAI*.

Shimon Kogan, Dimitry Levin, Bryan R. Routledge, Jacob S. Sagi, and Noah A. Smith. 2009. Predicting risk from financial reports with regression. In *Proc. of HLT-NAACL*.

Matthieu Kowalski and Bruno Torresani. 2009. Sparsity and persistence: mixed norms provide simple signal models with dependent coefficients. *Signal, Image and Video Processing*, 3(3):251–0264.

Xiaolei Lv, Guoan Bi, and Chunru Wan. 2011. The group lasso for stable recovery of block-sparse signal representations. *IEEE Transactions on Signal Processing*, 59(4):1371–1382.

Andre F. T. Martins, Noah A. Smith, Pedro M. Q. Aguiar, and Mario A. T. Figueiredo. 2011. Structured sparsity in structured prediction. In *Proc. of EMNLP*.

Anil Nelakanti, Cedric Archambeau, Julien Mairal, Francis Bach, and Guillaume Bouchard. 2013. Structured penalties for log-linear language models. In *Proc. of EMNLP*.

Kamal Nigam, Andrew McCallum, Sebastian Thrun, and Tom Mitchell. 2000. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39(2-3):103–134.

Bo Pang and Lilian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1–2):1–135.

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

M. J. D. Powell. 1969. A method for nonlinear constraints in minimization problems. In R. Fletcher, editor, *Optimization*, pages 283–298. Academic Press.

Zhiwei (Tony) Qin and Donald Goldfarb. 2012. Structured sparsity via alternating direction methods. *Journal of Machine Learning Research*, 13:1435–1468.

Dragomir R. Radev, Pradeep Muthukrishnan, and Vahed Qazvinian. 2009. The ACL anthology network corpus. In *Proc. of ACL Workshop on Natural Language Processing and Information Retrieval for Digital Libraries*.

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

Jiang Su, Jelber Sayyad-Shirabad, and Stan Matwin. 2011. Large scale text classication using semi-supervised multinomial naive Bayes. In *Proc. of ICML*.

Matt Thomas, Bo Pang, and Lilian Lee. 2006. Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *Proc. of EMNLP*.

Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. 2005. Sparsity and smoothness via the fused lasso. *Journal of Royal Statistical Society B*, 67(1):91–108.

Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of Royal Statistical Society B*, 58(1):267–288.

Tae Yano, Noah A. Smith, and John D. Wilkerson. 2012. Textual predictors of bill survival in congressional committees. In *Proc. of NAACL*.

Ainur Yessenalina, Yisong Yue, and Claire Cardie. 2010. Multi-level structured models for document sentiment classification. In *Proc. of EMNLP*.

Dani Yogatama and Noah A. Smith. 2014. Making the most of bag of words: Sentence regularization with alternating direction method of multipliers. In *Proc. of ICML*.

Dani Yogatama, Michael Heilman, Brendan O'Connor, Chris Dyer, Bryan R. Routledge, and Noah A. Smith. 2011. Predicting a scientific community's response to an article. In *Proc. of EMNLP*.

Ming Yuan and Yi Lin. 2006. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68(1):49–67.

Lei Yuan, Jun Liu, and Jieping Ye. 2013. Efficient methods for overlapping group lasso. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(9):2104–2116.

Hui Zou and Trevor Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320.

Perplexity on Reduced Corpora

Hayato Kobayashi*
Yahoo Japan Corporation
9-7-1 Akasaka, Minato-ku, Tokyo 107-6211, Japan
hakobaya@yahoo-corp.jp

Abstract

This paper studies the idea of removing low-frequency words from a corpus, which is a common practice to reduce computational costs, from a theoretical standpoint. Based on the assumption that a corpus follows Zipf's law, we derive trade-off formulae of the perplexity of k-gram models and topic models with respect to the size of the reduced vocabulary. In addition, we show an approximate behavior of each formula under certain conditions. We verify the correctness of our theory on synthetic corpora and examine the gap between theory and practice on real corpora.

1 Introduction

Removing low-frequency words from a corpus (often called *cutoff*) is a common practice to save on the computational costs involved in learning language models and topic models. In the case of language models, we often have to remove low-frequency words because of a lack of computational resources, since the feature space of k-grams tends to be so large that we sometimes need cutoffs even in a distributed environment (Brants et al., 2007). In the case of topic models, the intuition is that low-frequency words do not make a large contribution to the statistics of the models. Actually, when we try to roughly analyze a corpus with topic models, a reduced corpus is enough for the purpose (Steyvers and Griffiths, 2007).

A natural question arises: How many low-frequency words can we remove while maintaining sufficient performance? Or more generally, by how much can we reduce a corpus/model using a certain strategy and still keep a sufficient level of performance? There have been many studies addressing the question as it pertains to different strategies (Stolcke, 1998; Buchsbaum et al., 1998; Goodman and Gao, 2000; Gao and Zhang, 2002; Ha et al., 2006; Hirsimaki, 2007; Church et al., 2007). Each of these studies experimentally discusses trade-off relationships between the size of the reduced corpus/model and its performance measured by perplexity, word error rate, and other factors. To our knowledge, however, there is no theoretical study on the question and no evidence for such a trade-off relationship, especially for topic models.

In this paper, we first address the question from a theoretical standpoint. We focus on the cutoff strategy for reducing a corpus, since a cutoff is simple but powerful method that is worth studying; as reported in (Goodman and Gao, 2000; Gao and Zhang, 2002), a cutoff is competitive with sophisticated strategies such as entropy pruning. As the basis of our theory, we assume Zipf's law (Zipf, 1935), which is an empirical rule representing a long-tail property of words in a corpus. Our approach is essentially the same as those in physics, in the sense of constructing a theory while believing experimentally observed results. For example, we can derive the distance to the landing point of a ball thrown up in the air with initial speed v_0 and angle θ as ${v_0}^2 \sin(2\theta)/g$ by believing in the experimentally observed gravity acceleration g. In a similar fashion, we will try to clarify the trade-off relationship by believing Zipf's law.

The rest of the paper is organized as follows. In Section 2, we define the notation and briefly explain Zipf's law and perplexity. In Section 3, we theoretically derive the trade-off formulae of the cutoff for unigram models, k-gram models, and topic models, each of which represents its perplexity with respect to a reduced vocabulary, under the assumption that the corpus follows Zipf's law. In addition, we show an approximate behavior of each formula under certain conditions. In

*This work was mainly carried out while the author was with Toshiba Corporation.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 797–806,
Baltimore, Maryland, USA, June 23-25 2014.

Section 4, we verify the correctness of our theory on synthetic corpora and examine the gap between theory and practice on several real corpora. Section 5 concludes the paper.

2 Preliminaries

Let us consider a corpus $\mathbf{w} := w_1 \cdots w_N$ of corpus size N and vocabulary size W. We use an abridged notation $\{\mathbf{w}\} := \{w \in \mathbf{w}\}$ to represent the vocabulary of $\mathbf{w}$. Clearly, $N = |\mathbf{w}|$ and $W = |\{\mathbf{w}\}|$ hold. When $\mathbf{w}$ has additional notations, N and W inherit them. For example, we will use N' as the size of $\mathbf{w}'$ without its definition.

2.1 Power law and Zipf's law

A power law is a mathematical relationship between two quantities x and y, where y is proportional to the c-th power of x, i.e., $y \propto x^c$, and c is a real number. Zipf's law (Zipf, 1935) is a power law discovered on real corpora, wherein for any word $w \in \mathbf{w}$ in a corpus $\mathbf{w}$, its frequency (or word count) $f(w)$ is inversely proportional to its frequency ranking $r(w)$, i.e.,

$$f(w) = \frac{C}{r(w)}.$$

Here, $f(w) := |\{w' \in \mathbf{w} \mid w' = w\}|$, and $r(w) := |\{w' \in \mathbf{w} \mid f(w') \geq f(w)\}|$. From the definition, the constant C is the maximum frequency in the corpus. Taking the natural logarithms $\ln(\cdot)$ of both sides of the above equation, we find that its plot becomes linear on a log-log graph of $r(w)$ and $f(w)$. In fact, the result based on a statistical test in (Clauset et al., 2009) reports that the frequencies of words in a corpus completely follow a power law, whereas many datasets with long-tail properties, such as networks, actually do not follow power laws.

2.2 Perplexity

Perplexity is a widely used evaluation measure of k-gram models and topic models. Let p be a predictive distribution over words, which was learned from a training corpus $\mathbf{w}$ based on a certain model. Formally, perplexity *PP* is defined as the geometric mean of the inverse of the per-word likelihood on the held-out test corpus $\mathbf{w}_\tau$, i.e.,

$$PP := \left(\prod_{w \in \mathbf{w}_\tau} \frac{1}{p(w)} \right)^{\frac{1}{N_\tau}}.$$

Intuitively, *PP* means how many possibilities one has for estimating the next word in a test corpus. According to the definition, a lower perplexity means better generalization performance of p. Another well-known evaluation measure is cross-entropy. Since cross-entropy is easily calculated as $\log_2 PP$, we can apply many of the results of this paper to cross-entropy.

3 Perplexity on Reduced Corpora

Now let us consider what a cutoff is. In our study, we simply define a corpus that has been reduced by removing low-frequency words from the original corpus with a certain threshold. Formally, we say $\mathbf{w}'$ is a *corpus reduced from the original corpus* $\mathbf{w}$, if $\mathbf{w}'$ is the longest subsequence of $\mathbf{w}$ such that $\max_{w' \in \mathbf{w}'} r(w') = W'$. Note that a subsequence can include gaps in contrast to a substring. For example, supposing we have a corpus $\mathbf{w} = abcaba$ with a vocabulary $\{\mathbf{w}\} = \{a, b, c\}$, $\mathbf{w}'_1 = ababa$ is a reduced corpus, while $\mathbf{w}'_2 = aba$ and $\mathbf{w}'_3 = acaa$ are not.

After learning a distribution p' from a reduced corpus $\mathbf{w}'$, we need to infer the distribution p learned from the original corpus $\mathbf{w}$. Here, we use *constant restoring* (defined below), which assumes the frequencies of the reduced low-frequency words are a constant.

Definition 1 (Constant Restoring)**.** *Given a positive constant λ, a distribution p' over a reduced corpus $\mathbf{w}'$, and a corpus $\mathbf{w}$, we say that $\hat{p}$ is a λ-*restored distribution of p' from $\mathbf{w}'$ to $\mathbf{w}$*, if $\sum_{w \in \{\mathbf{w}\}} \hat{p}(w) = 1$, and for any $w \in \mathbf{w}$,*

$$\hat{p}(w) \propto \begin{cases} p'(w) & (w \in \mathbf{w}') \\ \lambda & (w \notin \mathbf{w}'). \end{cases}$$

Constant restoring is similar to the additive smoothing defined by $\hat{p}(w) \propto p'(w) + \lambda$, which is used to solve the zero-frequency problem of language models (Chen and Goodman, 1996). The only difference is the addition of a constant λ only to zero-frequency words. We think constant restoring is theoretically natural in our setting, since we can derive the above equation by letting each frequency of reduced words be $\lambda N'$ and defining a restored frequency function as follows:

$$\hat{f}(w) = \begin{cases} f(w) & (w \in \mathbf{w}') \\ \lambda N' & (w \notin \mathbf{w}'). \end{cases}$$

Informally, constant restoring involves padding the vocabulary, while additive smoothing involves padding the corpus. Smoothing should be carried out after restoring.

3.1 Perplexity of Unigram Models

Let us consider the perplexity of a unigram model learned from a reduced corpus. In unigram models, a predictive distribution p' on a reduced corpus $\mathbf{w}'$ can be simply calculated as $p'(w') = f(w')/N'$. We shall start with an analysis of training-set perplexity, since we can derive an exact formula for it, which will give us a sufficient idea for making an approximate analysis of test-set perplexity.

Let $\hat{PP}_1 := \left(\prod_{w\in\mathbf{w}} \frac{1}{\hat{p}(w)}\right)^{\frac{1}{N}}$ be the perplexity of a λ-restored distribution $\hat{p}$ on a unigram model. The next lemma gives the optimal restoring constant λ^* minimizing $\hat{PP}_1$.

Lemma 2. *For any λ-restored distribution $\hat{p}$ of a distribution p' from a reduced corpus $\mathbf{w}'$ to the original corpus $\mathbf{w}$, its perplexity is minimized by*

$$\lambda^* = \frac{N - N'}{(W - W')N'}.$$

Proof. Let $\mathbf{w}_\mathcal{R}$ be the longest subsequence such that $\min_{w'\in\mathbf{w}'} r(w') = W' + 1$. Since $\mathbf{w}_\mathcal{R}$ is the remainder of $\mathbf{w}'$, $N_\mathcal{R} = N - N'$ and $W_\mathcal{R} = W - W'$ hold. After substituting the normalized form of $\hat{p}$ of Definition 1 into $\hat{PP}_1$, we have

$$\begin{aligned}\hat{PP}_1 &= \left(\prod_{w'\in\mathbf{w}'} \frac{1}{\hat{p}(w')} \prod_{w_\mathcal{R}\in\mathbf{w}_\mathcal{R}} \frac{1}{\hat{p}(w_\mathcal{R})}\right)^{\frac{1}{N}} \\ &= \left(\prod_{w'\in\mathbf{w}'} \frac{1 + W_\mathcal{R}\lambda}{p'(w')} \prod_{w_\mathcal{R}\in\mathbf{w}_\mathcal{R}} \frac{1 + W_\mathcal{R}\lambda}{\lambda}\right)^{\frac{1}{N}} \\ &= \frac{1 + W_\mathcal{R}\lambda}{\lambda^{\frac{N_\mathcal{R}}{N}}} \left(\prod_{w'\in\mathbf{w}'} \frac{1}{p'(w')}\right)^{\frac{1}{N}}.\end{aligned}$$

We obtain the optimal smoothing factor λ^* when $\frac{\partial}{\partial\lambda}\hat{PP}_1 \propto \frac{\partial}{\partial\lambda}(1 + W_\mathcal{R}\lambda)/\lambda^{\frac{N_\mathcal{R}}{N}} = 0$. □

By using a similar argument to the one in the above lemma, we can obtain the optimal constant of additive smoothing as $\lambda^* \approx \frac{N-N'}{WN'}$, when N is sufficiently large.

The next theorem gives the exact formula of the training-set perplexity of a unigram model learned from a reduced corpus.

Theorem 3. *For any distribution p' on a unigram model learned from a corpus $\mathbf{w}'$ reduced from the original corpus $\mathbf{w}$ following Zipf's law, the perplexity $\hat{PP}_1$ of the λ^*-restored distribution $\hat{p}$ of p' from $\mathbf{w}'$ to $\mathbf{w}$ is calculated by*

$$\begin{aligned}\hat{PP}_1(W') =& H(W) \exp\left(\frac{B(W')}{H(W)}\right) \\ & \left(\frac{W - W'}{H(W) - H(W')}\right)^{1 - \frac{H(W')}{H(W)}},\end{aligned}$$

where $H(X) := \sum_{x=1}^{X} \frac{1}{x}$ and $B(X) := \sum_{x=1}^{X} \frac{\ln x}{x}$.

Proof. We expand the first part of $\hat{PP}_1$ in the proof of Lemma 2 using λ^* as follows:

$$\begin{aligned}\frac{1 + W_\mathcal{R}\lambda^*}{{\lambda^*}^{\frac{N_\mathcal{R}}{N}}} &= \left(1 + \frac{N_\mathcal{R}}{N'}\right)\left(\frac{W_\mathcal{R}N'}{N_\mathcal{R}}\right)^{\frac{N_\mathcal{R}}{N}} \\ &= \left(\frac{N}{N'}\right)\left(\frac{(W - W')N'}{N - N'}\right)^{1 - \frac{N'}{N}}.\end{aligned}$$

The second part of $\hat{PP}_1$ is as follows:

$$\begin{aligned}\left(\prod_{w'\in\mathbf{w}'} \frac{1}{p'(w')}\right)^{\frac{1}{N}} &= \prod_{w'\in\{\mathbf{w}'\}} \left(\frac{1}{p'(w')}\right)^{\frac{f(w')}{N}} \\ &= \prod_{r=1}^{W'} \left(\frac{rN'}{C}\right)^{\frac{C}{rN}} \\ &= \prod_{r=1}^{W'} \left(\frac{N'}{C}\right)^{\frac{C}{rN}} \prod_{r=1}^{W'} r^{\frac{C}{rN}} \\ &= \left(\frac{N'}{C}\right)^{\frac{N'}{N}} \exp\left(\frac{C}{N}\sum_{r=1}^{W'} \frac{\ln r}{r}\right).\end{aligned}$$

We obtain the objective formula by putting the above two formulae together with $N = CH(W)$ and $N' = CH(W')$, which are derived from Zipf's law. □

The functions $H(X)$ and $B(X)$ are the X-th partial sum of the harmonic series and Bertrand series (special form), respectively. An approximation by definite integrals yields $H(X) \approx \ln X + \gamma$, where γ is the Euler-Mascheroni constant, and $B(X) \approx \frac{1}{2}\ln^2 X$. We may omit γ from the approximate analysis.

Now let us consider an approximate form of $\hat{PP}_1(W')$ in Theorem 3. For further discussion,

we define the last part of $\hat{PP}_1(W')$ as follows:

$$F(W, W') := \left(\frac{W - W'}{H(W) - H(W')}\right)^{1-\frac{H(W')}{H(W)}}.$$

Since $W' = \delta W$ holds for an appropriate ratio δ, we have

$$\begin{aligned} F(W, \delta W) &= \left(\frac{W - \delta W}{H(W) - H(\delta W)}\right)^{1-\frac{H(\delta W)}{H(W)}} \\ &\approx \left(\frac{W - \delta W}{\ln W - \ln(\delta W)}\right)^{1-\frac{\ln(\delta W)}{\ln W}} \\ &= \left(\frac{W(1-\delta)}{-\ln \delta}\right)^{\frac{-\ln \delta}{\ln W}} \\ &\to \frac{1}{\delta} \qquad (W \to \infty). \end{aligned}$$

Therefore, when W is sufficiently large, we can use $F(W, W') \approx \frac{W}{W'}$, since $F(W, \delta W) \approx \frac{1}{\delta}$ holds for any ratio $\delta : 0 < \delta < 1$. Using this fact, we obtain an approximate formula $\tilde{PP}_1$ of $\hat{PP}_1$ as follows:

$$\begin{aligned} \tilde{PP}_1(W') &= \ln W \exp\left(\frac{\ln^2 W'}{2 \ln W}\right) \frac{W}{W'} \\ &= \sqrt{W} \ln W \exp \frac{(\ln W' - \ln W)^2}{2 \ln W}. \end{aligned}$$

The complexity of $\tilde{PP}_1$ is quasi-polynomial, i.e., $\tilde{PP}_1(W') = O(W'^{\ln W'})$, which behaves as a quadratic function on a log-log graph. Since $\tilde{PP}_1(W')$ is convex, i.e., $\frac{\partial^2}{\partial W'^2}\tilde{PP}_1(W') > 0$, and its gradient $\frac{\partial}{\partial W'}\tilde{PP}_1(W')$ is zero when $W' = W$, we infer that low-frequency words may not largely contribute to the statistics.

Considering the special case of $W' = W$, we obtain the perplexity PP_1 of the unigram model learned from the original corpus $\mathbf{w}$ as

$$PP_1 = H(W) \exp\left(\frac{B(W)}{H(W)}\right) \approx \sqrt{W} \ln W.$$

Interestingly, PP_1 is approximately expressed as a simple elementary function of vocabulary size W. This suggests that models learned from corpora with the same vocabulary size theoretically have the same perplexity.

For the test-set perplexity, we assume that both the training corpus $\mathbf{w}$ and test corpus $\mathbf{w}_\tau$ are generated from the same distribution based on Zipf's law. This assumption is natural, considering the situation of an in-domain test or cross validation test. Let $\mathbf{w}_\tau{}'$ be the longest subsequence of $\mathbf{w}_\tau$ such that for any $w \in \mathbf{w}_\tau{}'$, $w \in \mathbf{w}'$ holds. Formally, we assume $p'(w) \approx p_\tau{}'(w)$ for any $w \in \mathbf{w}'_\tau$ when $W_\tau > W'$, where $p_\tau{}'$ is the true distribution over $\mathbf{w}_\tau{}'$. Using similar arguments to those of Lemma 2 and Theorem 3 for $\mathbf{w}_\tau$, we obtain an approximation formula for the test-set perplexity, where we simply substitute W and W' in the exact formula for the training-set perplexity with W_τ and $W_\tau{}'$, respectively. For simplicity, we will only consider training-set perplexity from now on, since we can make a similar argument for the test-set perplexity in the later analysis.

3.2 Perplexity of k-gram Models

Here, we will consider the perplexity of a k-gram model learned from a reduced corpus as a standard extension of a unigram model. Our theory only assumes that the corpus is generated on the basis of Zipf's law. Thus, we can use a simple model where k-grams are calculated from a random word sequence based on Zipf's law. This model seems to be stupid, since we can easily notice that the bigram "is is" is quite frequent, and the two bigrams "is a" and "a is" have the same frequency. However, the experiments described later uncovered the fact that the model can roughly capture the behavior of real corpora.

The frequency f_k of k-gram word $w_k \in \mathbf{w}^k$ in the model is represented by the following formula:

$$f_k(w_k) = \frac{C_k}{g_k(r_k(w_k))},$$

where C_k is the maximal frequency in k-grams, r_k is the frequency ranking of w_k over k-grams, and g_k expresses the frequency decay in k-grams. For example, the decay function g_2 of bigrams is as follows:

$$\begin{aligned} (g_2(i))_i &:= (g_2(1),\ g_2(2),\ g_2(3), \cdots) \\ &= (1 \cdot 1,\ 1 \cdot 2,\ 2 \cdot 1,\ 1 \cdot 3,\ 3 \cdot 1, \cdots) \\ &= (1,\ 2,\ 2,\ 3,\ 3,\ 4,\ 4,\ 4,\ 5,\ 5,\ 6, \cdots). \end{aligned}$$

This is an inverse of the sum of Piltz's divisor functions $d_2(n) := \sum_{i_1 \cdot i_2 = n} 1$, which represents the number of divisors of an integer n (cf. (OEIS, 2001)). In general, we formally define g_k through its inverse: $g_k^{-1}(\ell) := S_k(\ell)$, where $S_k(\ell) := \sum_{n=1}^{\ell} d_k(n)$ and $d_k(n) := \sum_{i_1 \cdot i_2 \cdots i_k = n} 1$. Since $(g_k(i))_i$ is a sorted sequence of the elements of the k-th tensor power of vector $(1, \cdots, W)$, we can calculate the maximum frequency C_k as follows.

Lemma 4. *For any corpus* $\mathbf{w}$ *following Zipf's law, the maximum frequency of* k*-grams in our model is calculated by*

$$C_k = \frac{N-(k-1)D}{(H(W))^k},$$

where D *denotes the number of documents in* $\mathbf{w}$.

Proof. We use $\sum_{w_k} f_k(w_k) = C_k(\sum_w 1/r(w))^k$. □

The sum $S_k(\ell)$ of Piltz's divisor functions can be approximated by $\ell P_k(\ln \ell)$, where $P_k(x)$ is a polynomial of degree $k-1$ with respect to x, and the main term of $\ell P_k(\ln \ell)$ is given by the following residue $\mathrm{Res}_{s=1}\frac{\zeta^k(s)x^s}{s}$, where $\zeta(s)$ is the Riemann zeta function (Li, 2005). Using this fact, we obtain an approximation $\ln\left(g_k^{-1}(\ell)\right) \approx \ln \ell + O(\ln(\ln \ell)) \approx \ln \ell$, when ℓ is sufficiently large. Thus, when the corpus is sufficiently large, we can see that the behavior of f_k is roughly linear on a log-log graph, i.e., $f_k(w_k) \propto r_k(w_k)^{-1}$, since if $g_k^{-1}(\ell) \propto \ell^c$ holds, then $f_k(r) \propto (g_k(r))^{-1} \propto r^{-\frac{1}{c}}$ holds.

Unfortunately, however, most corpora in the real world are not so large that the above-mentioned relation holds. Actually, Ha et al. (Ha et al., 2002; Ha et al., 2006) experimentally found that although a k-gram corpus roughly follows a power law even when $k > 1$, its exponent is smaller than 1 (for Zipf's law). They pointed out that the exponent of bigrams is about 0.66, and that of 5-grams is about 0.59 in the Wall Street Journal corpus (WSJ87). Believing their claim that there exists a constant π_k such that $f_k(w_k) \propto r_k(w_k)^{-\pi_k}$, we estimated the exponent of k-grams in an actual situation in the form of the following lemma.

Lemma 5. *Assuming that* $f_k(w_k) \propto r_k(w_k)^{-\pi_k}$ *holds for any* k*-gram word* $w_k \in \mathbf{w}^k$ *in a corpus* $\mathbf{w}$ *following Zipf's law, the optimal exponent in our model based on the least squares criterion is calculated by*

$$\pi_k = \frac{\ln W}{(k-1)\ln(\ln W) + \ln W}.$$

Proof. We find the optimal exponent π_k by minimizing the sum of squared errors between the gradients of $g_k^{-1}(r)$ and $r^{\frac{1}{\pi_k}}$ on a log-log graph:

$$\int \left\{ \frac{\partial}{\partial y}\left(y + \ln P_k(y)\right) - \frac{\partial}{\partial y}\left(\frac{1}{\pi_k}y\right)\right\}^2 dy,$$

where $y = \ln r$. □

In the case of unigrams ($k = 1$), the formula exactly represents Zipf's law. In the case of k-grams ($k > 1$), we found that the formula approaches Zipf's law when W approaches infinity, i.e., $\lim_{W\to\infty} \pi_k = 1$.

Let us consider the perplexity of a k-gram model learned from a reduced corpus. We immediately obtain the following corollary using Lemma 5.

Corollary 6. *For any distribution* p' *on a* k*-gram model learned from a corpus* $\mathbf{w}'$ *reduced from the original corpus* $\mathbf{w}$ *following Zipf's law, assuming that* $f_k(w_k) \propto r_k(w_k)^{-\pi_k}$ *holds for any* k*-gram word* $w_k \in \mathbf{w}^k$ *and the optimal exponent* π_k *in Lemma 5, the perplexity* $\hat{PP}_k$ *of the* λ^**-restored distribution* $\hat{p}$ *of* p' *from* $\mathbf{w}'$ *to* $\mathbf{w}$ *is calculated by*

$$\hat{PP}_k(W') = H_{\pi_k}(W)\exp\left(\frac{B_{\pi_k}(W')}{H_{\pi_k}(W)}\right) \left(\frac{W-W'}{H_{\pi_k}(W)-H_{\pi_k}(W')}\right)^{1-\frac{H_{\pi_k}(W')}{H_{\pi_k}(W)}},$$

where $H_a(X) := \sum_{x=1}^{X}\frac{1}{x^a}$ *and* $B_a(X) := \sum_{x=1}^{X}\frac{a\ln x}{x^a}$.

$H_a(X)$ is the X-th partial sum of the P-series or hyper-harmonic series, which is a generalization of the harmonic series $H(X)$. $B_a(X)$ is the X-th partial sum of the Bertrand series (another special form of $B(X)$). When $0 < a < 1$, we can easily calculate $\hat{PP}_k(W')$ by using the following approximations:

$$H_a(X) \approx \frac{(X+1)^{1-a}-1}{1-a}$$
$$B_a(X) \approx \frac{a}{1-a}(X+1)^{1-a}\ln(X+1) - \frac{a}{(1-a)^2}(X+1)^{1-a} + \frac{a}{(1-a)^2}.$$

By putting the approximations of $H_a(X)$ and $B_a(X)$ into the formula of Corollary 6, we obtain an approximation $\hat{PP}_k(W') \approx O({W'}^{{W'}^{1-\pi_k}})$. This implies that $\hat{PP}_k(W')$ is approximately linear on a log-log graph, when π_k is close to 1, i.e., k is relatively small and W is sufficiently large. Note that we must use the approximation of $H(X)$, not $H_a(X)$, when $a = 1$.

The fact that the frequency of k-grams follows a power law leads us to an additional convenient

property, since the process of generating a corpus in our theory can be treated as a variant of the coupon collector's problem. In this problem, we consider how many trials are needed for collecting all coupons whose occurrence probabilities follow some stable distribution. According to a well-known result about power law distributions (Boneh and Papanicolaou, 1996), we need a corpus of size $\frac{kW^k}{1-\pi_k}\ln W$ when $\pi_k < 1$, and $W\ln^2 W$ when $\pi_k = 1$ for collecting all of the k-grams, the number of which is W^k. Using results in (Atsonios et al., 2011), we can easily obtain a lower and upper bound of the actual vocabulary size $\tilde{W}_k$ of k-grams from the corpus size N and vocabulary size W as

$$\tilde{W}_k \geq (\pi_k + 1)\left(1 - e^{-\frac{(1-\pi_k)N}{W^k-1} - \ln\frac{W^k-1}{W^k}}\right)$$

$$\tilde{W}_k \leq \frac{\pi_k}{\pi_k - 1}\left(\frac{N}{H_{\pi_k}(W^k)}\right)^{\frac{1}{\pi_k}} - \frac{NW^{1-\pi_k}}{(\pi_k - 1)H_{\pi_k}(W^k)}.$$

This means that we can determine the rough sparseness of k-grams and adjust some of the parameters such as the gram size k in learning statistical language models.

3.3 Perplexity of Topic Models

In this section, we consider the perplexity of the widely used topic model, Latent Dirichlet Allocation (LDA) (Blei et al., 2003), by using the notation given in (Griffiths and Steyvers, 2004). LDA is a probabilistic language model that generates a corpus as a mixture of hidden topics, and it allows us to infer two parameters: the document-topic distribution θ that represents the mixture rate of topics in each document, and the topic-word distribution ϕ that represents the occurrence rate of words in each topic. For a given corpus $\mathbf{w}$, the model is defined as

$$\begin{aligned} \theta_{d_i} &\sim \textit{Dirichlet}(\alpha) \\ z_i|\theta_{d_i} &\sim \textit{Multi}(\theta_{d_i}) \\ \phi_{z_i} &\sim \textit{Dirichlet}(\beta) \\ w_i|z_i, \phi_{z_i} &\sim \textit{Multi}(\phi_{z_i}), \end{aligned}$$

where d_i and z_i are respectively the document that includes the i-th word w_i and the hidden topic that is assigned to w_i. In the case of inference by Gibbs sampling presented in (Griffiths and Steyvers, 2004), we can sample a "good" topic assignment z_i for each word w_i with high probability. Using the assignments $\mathbf{z}$, we obtain the posterior distributions of two parameters as $\hat{\theta}_d(z) \propto n_z^{(d)} + \alpha$ and $\hat{\phi}_z(w) \propto n_z^{(w)} + \beta$, where $n_z^{(d)}$ and $n_z^{(w)}$ respectively represent the number of times assigning topic z in document d and the number of times topic z is assigned to word w.

Since an exact analysis is very hard, we will place rough assumptions on $\hat{\phi}$ and $\hat{\theta}$ to reduce the complexity. The assumption placed on $\hat{\phi}$ is that the word distribution $\hat{\phi}_z$ of each topic z follows Zipf's law. We think this is acceptable since we can regard each topic as a corpus that follows Zipf's law. Since $\hat{\phi}_z$ is normalized for each topic, we can assume that for any two topics, z and z', and any two words, w and w', $\hat{\phi}_z(w) \approx \hat{\phi}_{z'}(w')$ holds if $r_z(w) = r_{z'}(w')$, where $r_z(w)$ is the frequency ranking of w with respect to $n_z^{(w)}$. Note that the above assumption pertains to a posterior, and we do not discuss the fact that a Pitman-Yor process prior is better suited for a power law (Goldwater et al., 2011).

The assumption placed on $\hat{\phi}$ may not be reasonable in the case of $\hat{\theta}$, because we can easily think of a document with only one topic, and we usually use a small number T of topics for LDA, e.g., $T = 20$. Thus, we consider two extreme cases. One is where each document evenly has all topics, and the other is where each document only has one topic. Although these two cases might be unrealistic, the actual (theoretical) perplexity is expected to be between their values. We believe that analyzing such extreme cases is theoretically important, since it would be useful for bounding the computational complexity and predictive performance.

We can regard the former case as a unigram model, since the marginal predictive distribution $\sum_{z=1}^{T}\hat{\theta}_d(z)\hat{\phi}_z(w) \propto \sum_{z=1}^{T}\frac{n_z^{(w)}+\beta}{T} \propto\!\!\propto f(w)$ is independent of d; here we have used $\hat{\theta}_d(z) = 1/T$ from the assumption. In the latter case, we can obtain an exact formula for the perplexity of LDA when the topic assigned to each document follows a discrete uniform distribution, as shown in the next theorem. Note that a mixture of corpora following Zipf's law can be approximately regarded as following Zipf's law, when W is sufficiently large.

Theorem 7. *For any distribution p' on the LDA model with T topics learned from a corpus $\mathbf{w}'$ reduced from the original corpus $\mathbf{w}$ following Zipf's law, assuming that each document only has one topic which is assigned based on a discrete uniform distribution, the perplexity $\hat{PP}_{Mix}$ of the λ^*-restored distribution $\hat{p}$ of p' from $\mathbf{w}'$ to $\mathbf{w}$ is calcu-*

Table 1: Details of Reuters, 20news, Enwiki, Zipf1, and ZipfMix.

	vocab. size	corpus size	doc. size
Reuters	70,258	2,754,800	18,118
20news	192,667	4,471,151	19,997
Enwiki	409,902	16,711,226	51,231
Zipf1	69,786	2,754,800	18,118
ZipfMix	70,093	2,754,800	18,118

lated by

$$\hat{PP}_{Mix}(W') = H(W/T)\exp\left(\frac{B(W'/T)}{H(W/T)}\right)\left(\frac{W-W'}{H(W/T)-H(W'/T)}\right)^{1-\frac{H(W'/T)}{H(W/T)}}$$

Proof. We can prove this by using a similar argument to that of Theorem 3 for each topic. □

The formula of the theorem is nearly identical to the one of Theorem 3 for a $1/T$ corpus. This implies that the growth rate of the perplexity of LDA models is larger than that of unigram models, whereas the perplexity of LDA models for the original corpus is smaller than that of unigram models. In fact, a similar argument to the one in the approximate analysis in Section 3.1 leads to an approximate formula $\tilde{PP}_{Mix}$ of $\hat{PP}_{Mix}$ as

$$\tilde{PP}_{Mix}(W') = \sqrt{\frac{W}{T}}\ln\frac{W}{T}\exp\frac{(\ln W' - \ln W)^2}{2\ln(W/T)},$$

when W is sufficiently large. That is, $\tilde{PP}_{Mix}(W')$ also has a quadratic behavior in a log-log graph, i.e., $\tilde{PP}_{Mix}(W') = O(W'^{\ln W'})$.

4 Experiments

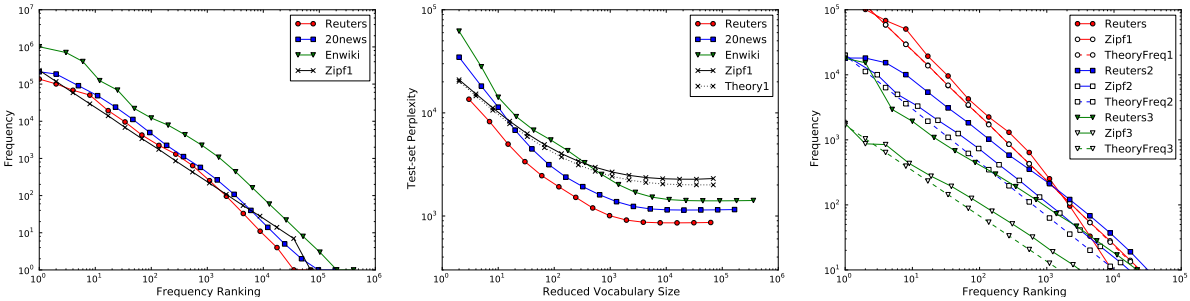
We performed experiments on three real corpora (Reuters, 20news, and Enwiki) and two synthetic corpora (Zipf1 and ZipfMix) to verify the correctness of our theory and to examine the gap between theory and practice. Reuters and 20news here denote corpora extracted from the Reuters-21578 and 20 Newsgroups data sets, respectively. Enwiki is a $1/100$ corpus of the English Wikipedia. Zipf1 is a synthetic corpus generated by Zipf's law, whose corpus is the same size as Reuters, and ZipfMix is a mixture of 20 synthetic corpora, sizes are 1/20th of Reuters. We used ZipfMix only for the experiments on topic models. Table 1 lists the details of all five corpora.

Fig. 1(a) shows the word frequency of Reuters, 20news, Enwiki, and Zipf1 versus frequency ranking on a log-log graph. In all corpora, we can regard each curve as linear with a gradient close to 1. This means that all corpora roughly follow Zipf's law. Furthermore, since the curve of Zipf1 is similar to that of Reuters, Zipf1 can be regarded as acceptable.

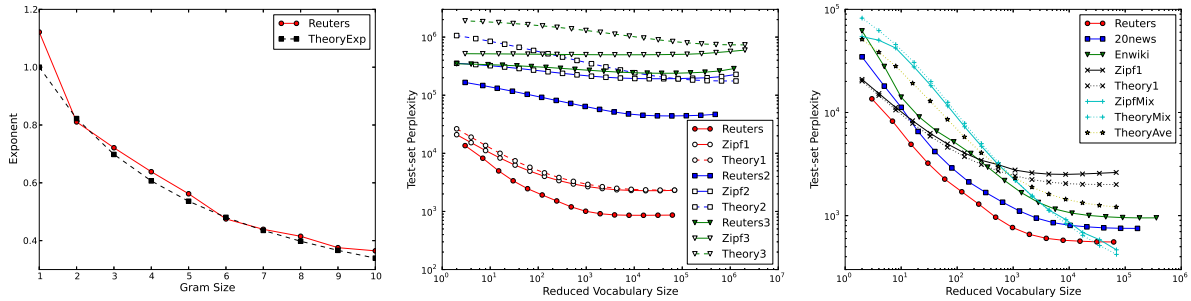
Fig. 1(b) plots the perplexity of unigram models learned from Reuters, 20news, Enwiki, and Zipf1 versus the size of reduced vocabulary on a log-log graph. Each value is the average over different test sets of five-fold cross validation. Theory1 is calculated using the formula in Theorem 3. The graph shows that the curve of Theory1 is nearly identical to that of Zipf1. Since the vocabulary size W_τ of each test set is small in this experiment, some errors appear when W' is large, i.e., $W_\tau < W'$. This clearly means that our theory is theoretically correct for an ideal corpus Zipf1. Comparing Zipf1 with Reuters, however, we find that their perplexities are quite different. The reason is that the gap between the frequencies of low-ranking (high-frequency) words is considerably large. For example, the frequency of the 1st-rank word of Reuters is $f(w) = 136,371$, while that of Zipf1 is $f(w) = 234,705$. Our theory seems to be suited for inferring the growth rate of perplexity rather than the perplexity value itself.

As for the approximate formula $\tilde{PP}_1$ of Theorem 3, we can surely regard the curve of Zipf1 as being roughly quadratic. The curves of real corpora also have a similar tendency, although their gradients are slightly steeper. This difference might have been caused by the above-mentioned errors. However, at least, we can ascertain the important fact that the results for the corpora reduced by 1/100 are not so different from those of the original corpora from the perspective of their perplexity measures.

Fig. 1(c) plots the frequency of k-grams ($k \in \{1,2,3\}$) in Reuters versus frequency ranking on a log-log graph. TheoryFreq (1-3) are calculated using C_k in Lemma 4 and π_k in Lemma 5. A comparison of TheoryFreq and Zipf verifies the correctness of our theory. However, comparing Zipf and Reuters, we see that C_k is poorly estimated when the gram size is large, whereas π_k is roughly correct. This may have happened because we did not put any assumptions on the word se-



(a) Frequency of unigrams (b) Perplexity of unigram models (c) Frequency of k-grams



(d) Exponent of a power law over k-grams (e) Perplexity of k-gram models (f) Perplexity of topic models

Figure 1: (a) Word frequency of `Reuters`, `20news`, `Enwiki`, and `Zipf1` versus frequency ranking. (b) Perplexity of unigram models learned from `Reuters`, `20news`, `Enwiki`, and `Zipf1` versus size of reduced vocabulary. `Theory1` is calculated using the formula in Theorem 3. (c) Frequency of k-grams ($k \in \{1, 2, 3\}$) in `Reuters` and `Zipf1` versus frequency ranking. The suffix digit of each label means its gram size. `TheoryFreq` (1-3) are calculated using Lemma 4 and Lemma 5. (d) Exponent of a power law over k-grams in `Reuters` versus gram size. `TheoryGrad` is calculated using π_k in Lemma 5. (e) Perplexity of k-gram models learned from `Reuters` versus size of reduced vocabulary. `Theory2` and `Theory3` are calculated using the formula in Corollary 6. (f) Perplexity of topic models learned from `Reuters`, `20news`, `Enwiki`, `Zipf1`, and `ZipfMix` versus size of reduced vocabulary. `TheoryMix` is calculated using the formula in Theorem 7.

quences in our simple model. The frequencies of high-order k-grams tend to be lower than in reality. We might need to place a hierarchical assumption on the a power law, as in done in hierarchical Pitman-Yor processes (Wood et al., 2011).

Fig. 1(d) plots the exponent of the power law over k-grams in `Reuters` versus the gram size on a normal graph. We estimated each exponent of `Reuters` by using the least-squares method. `TheoryGrad` is calculated using π_k in Lemma 5. Surprisingly, the real exponents of `Reuters` are almost the same as the theoretical estimate π_k based on our "stupid" model that does not care about the order of words. Note that we do not use any information other than the vocabulary size W and the gram size k for estimating π_k.

Fig. 1(e) plots the perplexity of k-gram models ($k \in \{1, 2, 3\}$) learned from `Reuters` versus the size of reduced vocabulary on a log-log graph. `Theory2` and `Theory3` are calculated using the formula in Corollary 6. In the case of bigrams, the perplexities of `Theory2` are almost the same as that of `Zipf2` when the size of reduced vocabulary is large. However, in the case of trigrams, the perplexities of `Theory3` are far from those of `Zipf3`. This difference may be due to the sparseness of trigrams in `Zipf3`. To verify the correctness of our theory for higher order k-gram models, we need to make assumptions that include backoff and smoothing.

Fig. 1(f) plots the perplexity of LDA models with 20 topics learned from `Reuters`, `20news`, `Enwiki`, `Zipf1`, and `ZipfMix` versus the size of reduced vocabulary on a log-log graph. We used a collapsed Gibbs sampler with 100 iterations to infer the parameters and set the hyper parameters, $\alpha = 0.1$ and $\beta = 0.1$. In evaluating the perplexity, we estimated a posterior document-topic distribu-

Table 2: Computational time and memory size for LDA learning on the original corpus, (1/10)-reduced corpus, and (1/20)-reduced corpus of `Reuters`.

corpus	time	memory	perplexity
original	4m3.80s	71,548KB	500
(1/10)	3m55.70s	46,648KB	550
(1/20)	3m42.63s	34,024KB	611

tion $\hat{\theta}_d$ by using the first half of each test document and calculated the perplexity on the second half, as is done in (Asuncion et al., 2009). Each value is the average over different test sets of five-fold cross validation. `Theory1` and `TheoryMix` are calculated using the formulae in Theorem 3 and Theorem 7, respectively. Comparing `Zipf1` with `Theory1`, and `ZipfMix` with `TheoryMix`, we find that our theory of the extreme cases discussed in Section 3.3 is theoretically correct. `TheoryAve` is the average of `Theory1` and `TheoryMix`. Comparing `Reuters` and `TheoryAve`, we see that their curves are almost the same. If theoretical perplexity $\hat{PP}$ has a similar tendency as real perplexity PP on a log-log graph, i.e., $\ln PP(W') \approx \ln \hat{PP}(W') + c$ for some constant c, we can approximate its deterioration rate as $PP(W')/PP(W) \approx \exp(\ln \hat{PP}(W') + c)/\exp(\ln \hat{PP}(W) + c) = \hat{PP}(W')/\hat{PP}(W)$. Therefore, we can use `TheoryAve` as a heuristic function for estimating the perplexity of topic models. Since we can calculate an inverse of `TheoryAve` from the bisection or Newton-Raphson method, we can maximize the reduction rate and ensure an acceptable perplexity based on a user-specified deterioration rate. According to the fact that the three real corpora with different sizes have a similar tendency, it is expected that we can use our theory for a larger corpus.

Finally, let us examine the computational costs for LDA learning. Table 2 shows computational time and memory size for LDA learning on the original corpus, (1/10)-reduced corpus, and (1/20)-reduced corpus of `Reuters`. Comparing the memory used in the learning with the original corpus and with the (1/10)-reduced corpus of `Reuters`, we find that the learning on the (1/10)-reduced corpus used 60% of the memory used by the learning on the original corpus. While the computational time decreased a little, we believe that reducing the memory size helps to reduce computational time for a larger corpus in the sense that it can relax the constraint for in-memory computing. Although we did not examine the accuracy of real tasks in this paper, there is an interesting report that the word error rate of language models follows a power law with respect to perplexity (Klakow and Peters, 2002). Thus, we conjecture that the word error rate also has a similar tendency as perplexity with respect to the reduced vocabulary size.

5 Conclusion

We studied the relationship between perplexity and vocabulary size of reduced corpora. We derived trade-off formulae for the perplexity of k-gram models and topic models with respect to the size of reduced vocabulary and showed that each formula approximately has a simple behavior on a log-log graph under certain conditions. We verified the correctness of our theory on synthetic corpora and examined the gap between theory and practice on real corpora. We found that the estimation of the perplexity growth rate is reasonable. This means that we can maximize the reduction rate, thereby ensuring an acceptable perplexity based on a user-specified deterioration rate. Furthermore, this suggests the possibility that we can theoretically derive empirical parameters, or "rules of thumb", for different NLP problems, assuming that a corpus follows Zipf's law. We believe that our theoretical estimation has the advantages of computational efficiency and scalability especially for very large corpora, although experimental estimations such as cross-validation may be more accurate.

In the future, we want to find out the cause of the gap between theory and practice and extend our theory to bridge the gap, in the same way that we can construct equations of motion with air resistance in the example of the landing point of a ball in Section 1. For example, promising research directions include using a general law such as the Zipf-Mandelbrot law (Mandelbrot, 1965), a sophisticated model that cares the order of words such as hierarchical Pitman-Yor processes (Wood et al., 2011), and smoothing/backoff methods to handle the sparseness problem.

Acknowledgments

The author would like to thank the reviewers for their helpful comments.

References

Arthur Asuncion, Max Welling, Padhraic Smyth, and Yee Whye Teh. 2009. On smoothing and inference for topic models. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI2009)*, pages 27–34. AUAI Press.

Ioannis Atsonios, Olivier Beaumont, Nicolas Hanusse, and Yusik Kim. 2011. On power-law distributed balls in bins and its applications to view size estimation. In *Proceedings of the 22nd International Symposium on Algorithms and Computation (ISAAC 2011)*, pages 504–513. Springer-Verlag.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

Shahar Boneh and Vassilis G. Papanicolaou. 1996. General asymptotic estimates for the coupon collector problem. *Journal of Computational and Applied Mathematics*, 67(2):277–289.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large Language Models in Machine Translation. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL2007)*, pages 858–867. ACL.

Adam L. Buchsbaum, Raffaele Giancarlo, and Jeffery R. Westbrook. 1998. Shrinking Language Models by Robust Approximation. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 1998)*, pages 685–688.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics (ACL 1996)*, pages 310–318. ACL.

Ken Church, Ted Hart, and Jianfeng Gao. 2007. Compressing Trigram Language Models with Golomb Coding. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, pages 199–207. ACL.

Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman. 2009. Power-Law Distributions in Empirical Data. *SIAM Review*, 51(4):661–703.

Jianfeng Gao and Min Zhang. 2002. Improving Language Model Size Reduction using Better Pruning Criteria. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, pages 176–182. ACL.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2011. Producing Power-Law Distributions and Damping Word Frequencies with Two-Stage Language Models. *Journal of Machine Learning Research*, 12:2335–2382.

Joshua Goodman and Jianfeng Gao. 2000. Language Model Size Reduction by Pruning and Clustering. In *Proceedings of the 6th International Conference on Spoken Language Processing (ICSLP 2000)*, pages 110–113. ISCA.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. In *Proceedings of the National Academy of Sciences of the United States of America (PNAS 2004)*, volume 101, pages 5228–5235.

Le Quan Ha, E. I. Sicilia-Garcia, Ji Ming, and F. J. Smith. 2002. Extension of Zipf's Law to Words and Phrases. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*, pages 1–6. ACL.

Le Quan Ha, P. Hanna, D. W. Stewart, and F. J. Smith. 2006. Reduced n-gram models for English and Chinese corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (COLING-ACL 2006)*, pages 309–315. ACL.

Teemu Hirsimaki. 2007. On Compressing N-Gram Language Models. In *Proceedings of the 2007 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2007)*, pages 949–952.

Dietrich Klakow and Jochen Peters. 2002. Testing the correlation of word error rate and perplexity. *Speech Communication*, 38(1):19–28.

Hailong Li. 2005. On Generalized Euler Constants and an Integral Related to the Piltz Divisor Problem. *Šiauliai Mathematical Seminar*, 8:81–93.

Benoit B. Mandelbrot. 1965. Information Theory and Psycholinguistics: A Theory of Word Frequencies. In *Scientific Psychology: Principles and Approaches*. Basic Books.

OEIS. 2001. The on-line encyclopedia of integer sequences (a061017). `http://oeis.org/A061017/`.

Mark Steyvers and Tom Griffiths. 2007. Probabilistic Topic Models. In *Handbook of Latent Semantic Analysis*, pages 424–440. Lawrence Erlbaum Associates.

Andreas Stolcke. 1998. Entropy-based Pruning of Backoff Language Models. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, pages 270–274.

Frank Wood, Jan Gasthaus, Cédric Archambeau, Lancelot James, and Yee Whye Teh. 2011. The Sequence Memoizer. *Communications of the Association for Computing Machines*, 54(2):91–98.

George Kingsley Zipf. 1935. *The Psychobiology of Language*. Houghton-Mifflin.

Robust Domain Adaptation for Relation Extraction via Clustering Consistency

Minh Luan Nguyen[†*], **Ivor W. Tsang**[‡], **Kian Ming A. Chai**[§], **and Hai Leong Chieu**[§]

[†]Institute for Infocomm Research, Singapore
[‡]Centre for Quantum Computation & Intelligent Systems, University of Technology, Sydney, Australia
[§]DSO National Laboratories, Singapore

mlnguyen@i2r.a-star.edu.sg, ivor.tsang@gmail.com
{ckianmin,chaileon}@dso.org.sg

Abstract

We propose a two-phase framework to adapt existing relation extraction classifiers to extract relations for new target domains. We address two challenges: negative transfer when knowledge in source domains is used without considering the differences in relation distributions; and lack of adequate labeled samples for rarer relations in the new domain, due to a small labeled data set and imbalance relation distributions. Our framework leverages on both labeled and unlabeled data in the target domain. First, we determine the relevance of each source domain to the target domain for each relation type, using the consistency between the clustering given by the target domain labels and the clustering given by the predictors trained for the source domain. To overcome the lack of labeled samples for rarer relations, these clusterings operate on both the labeled and unlabeled data in the target domain. Second, we trade-off between using relevance-weighted source-domain predictors and the labeled target data. Again, to overcome the imbalance distribution, the source-domain predictors operate on the unlabeled target data. Our method outperforms numerous baselines and a weakly-supervised relation extraction method on ACE 2004 and YAGO.

1 Introduction

The World Wide Web contains information on real-world entities, such as persons, locations and organizations, which are interconnected by various semantic relations. Detecting these relations between two entities is important for many tasks on the Web, such as information retrieval (Salton and McGill, 1986) and information extraction for question answering (Etzioni et al., 2008). Recent work on relation extraction has demonstrated that supervised machine learning coupled with intelligent feature engineering can provide state-of-the-art performance (Jiang and Zhai, 2007b). However, most supervised learning algorithms require adequate labeled data for every relation type to be extracted. Due to the large number of relations among entities, it may be costly to annotate a large enough set of training data to cover each relation type adequately in every new domain of interest. Instead, it can be more cost-effective to adapt an existing relation extraction system to the new domain using a small set of labeled data. This paper considers relation adaptation, where a relation extraction system trained on many source domains is adapted to a new target domain.

There are at least three challenges when adapting a relation extraction system to a new domain. First, the same semantic relation between two entities can be expressed using different lexical or syntactic patterns. For example, the acquisition of company A by company B can be expressed with "B bought over by A", "A buys B" and "A purchases B". To extract a relation, we need to capture the different ways in which it can be expressed across different open domains on the Web.

Second, the emphasis or interest on the different relation types varies from domain to domain. For example, in the organization domain, we may be more interested in extracting relations such as *locatedIn* (between a company and a location) and *founderOf* (between a company and a person),

[*]The work is done while Nguyen was a research staff in Nanyang Technological University, Singapore.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 807–817,
Baltimore, Maryland, USA, June 23-25 2014.

whereas in the person domain we may be more interested in extracting relations such as *liveIn* (between a person and a location) and *workAt* (between a person and a company). Therefore, although the two domains may have the same set of relations, they probably have different marginal distributions on the relations. This can produce a negative transfer phenomenon (Rosenstein et al., 2005), where using knowledge from other domains degrades the performance on the target domain. Hence, when transferring knowledge from multiple domains, it is overly optimistic to believe that all source domains will contribute positively. We call a source domain *irrelevant* when it has no or negative contribution to the performance of the target domain. One example is named entities extraction adaptation, where naïve transfer of information from a mixed-case domain with capitalization information (e.g., news-wire) to a single-case domain (e.g., conversational speech transcripts) will miss most names in the single-case domain due to the absence of case information, which is typically important in the mixed-case domain.

Third, the annotated instances for the target domain are typically much fewer than those for the source domains. This is primarily due to the lack of resources such as raw target domain documents, time, and people with the expertise. Together with imbalanced relation distributions inherent in the domain, this can cause some rarer relations to constitute only a very small proportion of the labeled data set. This makes learning a relation classifier for the target domain challenging.

To tackle these challenges, we propose a two-phase Robust Domain Adaptation (RDA) framework. In the first phase, Supervised Voting is used to determine the relevance of each source domain to each region in the target domain, using both labeled and unlabeled data in the target domain. By using also unlabeled data, we alleviate the lack of labeled samples for rarer relations due to imbalanced distributions in relation types.

The second phase uses the relevances determined the first phase to produce a reference predictor by weighing the source-domain predictors for each target domain sample separately. The intention is to alleviate the effect of mismatched distributions. The final predictor in the target domain is trained on the labeled target domain data while taking reference from the reference predictions on the unlabeled target domain data. This ensures reasonable predictive performance even when all the source domains are irrelevant and augments the rarer classes with examples in the unlabeled data. We compare the proposed two-phase framework with state-of-the-art domain adaptation baselines for the relation extraction task, and we find that our method outperforms the baselines.

2 Related Work

Relation extraction is usually considered a classification problem: determine if two given entities in a sentence have a given relation. Kernel-based supervised methods such as dependency tree kernels (Culotta and Sorensen, 2004), subsequence kernels (Bunescu and Mooney, 2006) and convolution tree kernels (Qian et al., 2008) have been rather successful in learning this task. However, purely supervised relation extraction methods assume the availability of sufficient labeled data, which may be costly to obtain for new domains. We address this by augmenting a small labeled data set with other information in the domain adaptation setting.

Bootstrapping methods (Zhu et al., 2009; Agichtein and Gravano, 2000; Xu et al., 2010; Pasca et al., 2006; Riloff and Jones, 1999) to relation extraction are attractive because they require fewer training instances than supervised approaches. Bootstrapping methods are either initialized with a few instances (often designated as seeds) of the target relation (Zhu et al., 2009; Agichtein and Gravano, 2000) or a few extraction patterns (Xu et al., 2010). In subsequent iterations, new extraction patterns are discovered, and these are used to extract new instances. The quality of the extracted relations depends heavily on the seeds (Kozareva and Hovy, 2010). Different from bootstrapping, we not only use labeled target domain data as seeds, but also leverage on existing source-domain predictors to obtain a robust relation extractor for the target domain.

Open Information Extraction (Open IE) (Etzioni et al., 2008; Banko et al., 2008; Mesquita et al., 2013) is a domain-independent information extraction paradigm to extract relation tuples from collected corpus (Shinyama and Sekine, 2006) and Web (Etzioni et al., 2008; Banko et al., 2008). Open IE systems are initialized with a few domain-independent extraction patterns. To create labeled data, the texts are dependency-parsed, and the domain-independent patterns on the parses form the basis for extractions. Recently, to reduce

labeling effort for relation extraction, distant supervision (Mintz et al., 2009; Takamatsu et al., 2012; Min et al., 2013; Xu et al., 2013) has been proposed. This is an unsupervised approach that exploits textual features in large unlabeled corpora. In contrast to Open IE, we tune the relation patterns for a domain of interest, using labeled relation instances in source and target domains and unlabeled instances in the target domain.

Our work is also different from the multi-schema matching in database integration (Doan et al., 2003). Multi-schema matching finds relations between columns of schemas, which have the same semantic. In addition, current weighted schema matching methods do not address negative transfer and imbalance class distribution.

Domain adaptation methods can be classified broadly into weakly-supervised adaptation (Daume and Marcu, 2007; Blitzer et al., 2006; Jiang and Zhai, 2007a; Jiang, 2009), and unsupervised adaptation (Pan et al., 2010; Blitzer et al., 2006; Plank and Moschitti, 2013). In the weakly-supervised approach, we have plenty of labeled data for the source domain and a few labeled instances in the target domain; in the unsupervised approach, the data for the target domain are not labeled. Among these studies, Plank and Moschitti's is the closest to ours because it adapts relation extraction systems to new domains. Most other works focused on adapting from old to new relation types. Typical relation adaptation methods first identify a set of common features in source and target domains and then use those features as pivots to map source domain features to the target domain. These methods usually assume that each source domain is relevant to the task on the target domain. In addition, these methods do not handle the imbalanced distribution of relation data explicitly. In this work, we study how to learn the target prediction using only a few seed instances, while dealing with negative transfer and imbalanced relation distribution explicitly. These issues are seldom explored in relation adaptation.

3 Problem Statement

This section defines the domain adaptation problem and describes our feature extraction scheme.

3.1 Relation Extraction Domain Adaptation

Given two entities A and B in a sentence S, relation extraction is the task of selecting the relation y between A and B from a fixed set of c relation types, which includes the *not-a-relation* type. We introduce a feature extraction χ that maps the triple (A,B,S) to its feature vector x. Learning relation extraction can then be abstracted to finding a function p such that $p(\chi(A,B,S)) = p(x) = y$.

For adaptation, we have k source domains and a target domain. We shall assume that all domains have the same set of relation types. The target domain has a few labeled data $D_l = \{(x_i, y_i)\}_{i=1}^{n_l}$ and plenty of unlabeled data $D_u = \{(x_i)\}_{n_l+1}^{n_l+n_u}$, where n_l and n_u are the number of labeled and unlabeled samples respectively, x_i is the feature vector, y_i is the corresponding label (if available). Let $n = n_l + n_u$. For the sth source domain, we have an adequate labeled data set D^s. We define *domain adaptation* as the problem of learning a classifier p for relation extraction in the target domain using the data sets D_l, D_u and D^s, $s = 1,\ldots,k$.

3.2 Relational Feature Representation

We consider relation extraction as a classification problem, where each pair of entities A and B within a sentence S is a candidate relation instance. The contexts in which entities A and B co-occur provide useful features to the relations between them. We use the term context to refer a window of text in which two entities co-occur. A context might not necessarily be a complete sentence S. Retrieving contexts in which two entities co-occur has been studied in previous works investigating the relations between entities.

Given a pair of entities (A,B) in S, the first step is to express the relation between A and B with some feature representation using a feature extraction scheme χ. Lexical or syntactic patterns have been successfully used in numerous natural language processing tasks, including relation extraction. Jiang and Zhai (2007b) have shown that selected lexical and syntactic patterns can give good performance for relation extraction. Following their work[1], we also use lexical and syntactic patterns extracted from the contexts to represent the relations between entities. We extract features from a sequence representation and a parse tree representation of each relation instance. The details are as follows.

Entity Features Entity types and entity mention types are very useful for relation extraction. We

[1]The source code for extracting entity features is provided by the authors (Jiang and Zhai, 2007b).

use a subgraph in the relation instance graph (Jiang and Zhai, 2007b) that contains only the node presenting the head word of the entity A, labeled with the entity type or entity mention types, to describe a single entity attribute.

Sequence Features The sequence representation preserves the order of the tokens as they occur in the original sentence. Each node in the graph is a token augmented with its relevant attributes.

Syntactic Features The syntactic parse tree of the relation instance sentence can be augmented to represent the relation instance. Each node is augmented with relevant part-of-speech (POS) using the Python Natural Language Processing Tool Kit.

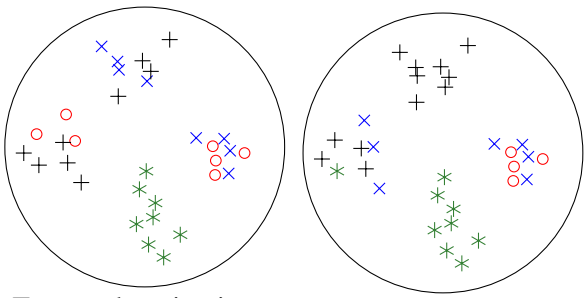
Each node in the sequence or the parse tree is augmented by an argument tag that indicates whether the node corresponds to entity A, B, both, or neither. The nodes that represent the argument are also labeled with the entity type, subtype and mention type. We trim the parse tree of a relation instance so that it contains only the most essential tree components based on constituent dependencies (Qian et al., 2008). We also use unigram features and bigram features from a relation instance graph.

4 Robust Domain Adaptation

In this section, we describe our two-phase approach, which comprises of a Supervised Voting scheme and a combined classifier learning phase.

4.1 Phase 1: Clustering Consistency via Supervised Voting

In this section, we use the concept of clustering consistency to determine the relevance of a source domain to particular regions in the target domain. Figure 1 illustrates this. There, both enclosing circles in the left and right figures denote the same input space of the target domain. There are four disjoint regions within the input space, located at the left, right, top and bottom of the space. There are four classes of labels: plus ($+$), cross ($\times$), circle ($\circ$) and asterisk ($*$). The labels in the left figure are given by a *preliminary predictor* in the target domain data, while the labels in the right figure are given by a predictor trained on the source domain data. Comparing the figures, we see the preliminary predictor and source domain predictor are consistent for the bottom and right regions,



Target domain input space with transductive learning using labeled and unlabeled target domain data.

Target domain input space with labels from the predictor trained on the source domain data set.

Figure 1: Clustering consistency is used to determine the relevance of a source domain to a region in the target domain data. The bottom and right regions are more relevant than the top and left regions. See text for explanation.

but are inconsistent for the top and left regions. This suggests that the source domain is very relevant for the bottom and right regions of the target input space, but less so for the top and left regions.

To apply this idea to relation classification, we have to (i) partition the target domain input space into regions and (ii) assign preliminary labels for all the examples. We approximate the target domain input space with all the samples from D_l and D_u. With data from both the labeled and unlabeled data sets, we apply transductive inference or semi-supervised learning (Zhou et al., 2003) to achieve both (i) and (ii). By augmenting with unlabeled data D_u, we aim to alleviate the effect of imbalanced relation distribution, which causes a lack of labeled samples for rarer classes in a small set of labeled data. Briefly, the known labels in D_l are propagated to the entire target input space by encouraging label smoothness in neighborhoods. The next three paragraphs give more details.

At present, we assume a similarity matrix W, where W_{ij} is the similarity between the ith and the jth input samples in $D_l \cup D_u$. Matrix W then determines the neighborhoods. Let Λ be a diagonal matrix where the (i,i)th entry is the sum of the ith row of W. Let us also encode the the labeled data D_l in an n-by-c matrix H, such that $H_{ij} = 1$ if sample i is labeled with relation class j in D_l, and $H_{ij} = 0$ otherwise. Our objective is the c-dimensional relation-class indicator vector F_i for the ith sample, for every sample. This is achieved

via a regularization framework (Zhou et al., 2003):

$$\min_{\{F_i\}_{i=1}^n} \left(\sum_{i,j=1}^{n} W_{ij} \left\| \frac{F_i}{\sqrt{\Lambda_{ii}}} - \frac{F_j}{\sqrt{\Lambda_{jj}}} \right\|^2 + \mu \sum_{i=1}^{n} \|F_i - H_i\|^2 \right).$$

This trades off two criteria: the first term encourages nearby samples (under distance metric W) to have the same labels, while the second encourages samples to take their labels from the labeled data. The closed-form solution is

$$F^* = (I - (1+\mu)^{-1} L)^{-1} H, \qquad (1)$$

where $L = \Lambda^{-1/2} W \Lambda^{-1/2}$; and the n-by-c matrix F^* is the concatenation of the F_is.

Using vector F_i^*, we now assign preliminary labels to the samples. For a sample i, we transform F_i^* into probabilities $p_i^1, p_i^2, \ldots, p_i^c$ using softmax. Our propagated label ℓ_i for sample i is then

$$\ell_i = \begin{cases} \textit{not-a-relation} & \text{if } (\max_j p_i^j) < \theta, \\ \arg\max_j p_i^j & \text{otherwise.} \end{cases} \qquad (2)$$

The second clause is self-evident, but the first needs further explanation. Because *not-a-relation* is a background or default relation type in the relation classification task, and because it has rather high variation when manifested in natural language, we have found it difficult to obtain a distance metric W that allows the *not-a-relation* samples to form clusters naturally using transductive inference. Therefore, we introduce the first clause to assign the *not-a-relation* label to a sample when there is no strong evidence for any of the positive relation types. The amount of evidence needed is quantified by the parameter $\theta > 1/c$. In addition, the second clause will also assign *not-a-relation* to a sample if that probability is the highest.

Next, we partition the data in $D_l \cup D_u$ into c regions, $R_1, R_2, \ldots, R_c$, corresponding to the c relation types. The intuition is to use the true label in D_l when available, or otherwise resort to using the propagated label. That is,

$$x_i \in \begin{cases} R_{y_i} & \text{if } x_i \in D_l, \\ R_{\ell_i} & \text{if } x_i \in D_u. \end{cases}$$

We now have the necessary ingredients to quantify the *clustering consistency* between a source domain and a region in the target domain. Intuitively, this is the agreement between the source-domain predictor and the preliminary predictor within the target domain. We use supervised voting in the following manner. For every source domain, say domain s, we first train a relation-type predictor p_s based on its training data D^s. Then, for every region R_j, we compute the relevance score $w_{s,j} = \sum_{x_i \in R_j} [\![p_s(x_i) = \ell_i]\!] / |R_j|$, where $[\![\cdot]\!]$ is the Iverson bracket.

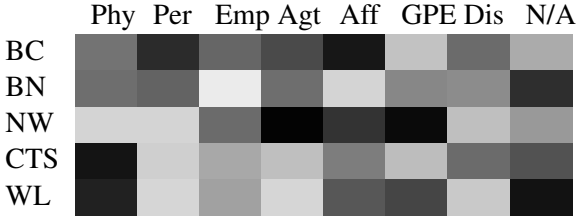


Figure 2: Heat map of the relevance scores $w_{s,j}$ between the target domain Usenet (UN) with the other domains on ACE 2004 data set. A lighter shade means a higher score, or more relevant. N/A refers to *not-a-relation*; for the other abbreviations, see the second paragraph in section 5.

Figure 2 shows the heat map of the relevance scores $w_{s,j}$ between the target domain Usenet (UN) with the other domains in the ACE 2004 corpus. We observe, for example, that the Broadcast News (BN) domain is more relevant in the Personal-Social region of the target domain than the Broadcast Conversation (BC) domain. These relevance scores will be used in the next phase of the framework to weigh the contributions of source-domain predictors to the eventual target-domain relation classifier.

4.2 Phase 2: Target Classifier Learning

The second phase uses both the weighted predictions from all sources and the target labeled data D_l to learn a relation classifier. This ensures that even when most of the source domains are irrelevant, the performance of our method is no worse than using the target-domain labeled data alone.

The previous phase has computed the relevance $w_{s,j}$ for source domain s in region R_j. We translate this to the relevance weight $u_{s,i}$ for an example x_i: if $x_i \in R_j$, then $u_{s,i} = w_{s,j}$. At our disposal from the previous phase are also k source-domain predictors p_s that have been trained on D^s. Combining and weighing the predictions from multiple sources, we obtain the *reference predic-*

tion $\hat{r}_{ji} = \sum_{s=1}^{k} u_{s,i}(2[\![p_s(x_i) = j]\!] - 1)$ for example x_i belonging to relation j, using the ± 1 encoding.

The relation classifier consists of c functions $f_1, \ldots, f_c$ using the one-versus-rest decoding for multi-class classification.[2] Inspired by the Domain Adaptive Machine (Duan et al., 2009), we combine the reference predictions and the labeled data of the target domain to learn these functions:

$$\min_{\{f_j\}_{j=1}^{c}} \sum_{j=1}^{c} \left\{ \frac{1}{n_l} \sum_{i=1}^{n_l} (f_j(x_i) - r_{ji})^2 + \gamma \|f_j\|_{\mathcal{H}}^2 + \frac{\beta}{2} \sum_{i=n_l+1}^{n} \|f_j(x_i) - \hat{r}_{ji}\|^2 \right\}, \quad (3)$$

where $r_{ji} = 2[\![y_i = j]\!] - 1$ is the ± 1 binary encoding for the i labeled sample belonging to relation j. Here, we have multiple objectives: the first term controls the training error; the second regularizes the complexity of the functions f_js in the Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$; and the third prefers the predicted labels of the unlabeled data D_l to be close to the reference predictions. The third term provides additional pseudo-training samples for the rarer relation classes, if these are available in D_u. Parameters β and γ govern the trade-offs between these objectives.

Let $K(\cdot,\cdot)$ be the reproducing kernel for $\mathcal{H}$. By the Representer Theorem (Smola and Scholkopf, 1998), the solution for Eq. 3 is linear in $K(x_i,\cdot)$: $f_j(x) = \sum_{i=1}^{n} \alpha_{ji} K(x_i, x)$. Putting this into Eq. 3, parameter vectors α_j are (Belkin et al., 2006):

$$\alpha_j^* = (JK + \gamma(n_l + \beta n_u)I)^{-1} J R_j. \quad (4)$$

Here, R_j is an $(n_l + n_u)$-vector, where $R_{ji} = r_{ji}$ if sample i belongs to the labeled set, and $R_{ji} = \hat{r}_{ij}$ if it belongs to the unlabeled set; and J is an $(n_l + n_u)$-by-$(n_l + n_u)$ diagonal matrix where the first n_l diagonal entries are ones and the rest are βs.

5 Experiments

We evaluate our algorithm on two corpora: Automatic Content Extraction (ACE) 2004 and YAGO[3]. Table 1 provides some statistics on them.

ACE 2004 consists of six domains: Broadcast Conversations (BC), Broadcast News (BN), Conversational Telephone Speech (CTS), Newswire (NW), Usenet (UN) and Weblog (WL). There are seven positive relation types: Physical (Phy), Personal/Social (Per), Employment/Membership/Subsidiary (Emp), Agent-Artifact (Agt), PER/ORG Affiliation (Aff), GPE Affiliation (GPE) and Discourse (Dis).

Table 1: Statistics on ACE 2004 and YAGO

Properties	ACE 2004	YAGO
# relation types	7	20
# candidate relations	48,625	68,822
# gold relations	4,296	2,000
# mentions per entity pair	6	11
% mentions with +ve relations	8.8%	21%

YAGO is an open information extraction data set. The relation types of YAGO are built from Wikipedia and WordNet, while the labeled text for YAGO is from Bollegala et al. (2011). It consists of twenty relation types such as *ceo_company*, *bornIn* and *isMarriedTo*, and each of them is considered as a domain in this work. YAGO is different from ACE 2004 in two aspects: there is less overlapping of topics, entity types and relation types between domains; and it has more relation mentions with 11 mentions per pair of entities on the average.

We used Collins parser (Collins, 1999) to parse the sentences. The constituent parse trees were then transformed into dependency parse trees, using the head of each constituent (Jiang and Zhai, 2007b). The candidate relation instances were generated by considering all pairs of entities that occur in the same sentence. For the similarity matrix W in section 4.1 and the kernel $K(\cdot,\cdot)$ in section 4.2, we used the composite kernel function (Zhang et al., 2006), which is based on structured features and entity-related features.

F_1 is used to measure the performance of the algorithms. This is the harmonic mean of precision $\mathrm{TP}/(\mathrm{TP}+\mathrm{FP})$ and recall $\mathrm{TP}/(\mathrm{TP}+\mathrm{FN})$, where TP, FP and FN are the numbers of correct, missing and wrongly recognized relations.

5.1 Experimental Settings

For ACE 2004, we used each of the six domains as the target domain and the remaining domains as source domains. For YAGO, each relation type in YAGO was considered as a domain. For each domain in YAGO, we have a binary classification task: whether the instance has the relation corresponding to the domain. Five-fold cross-validation was used to evaluate the performance.

[2] For two-classes, though, only one function is needed.

[3] http://www.mpi-inf.mpg.de/yago-naga/yago/

For every target domain, we divided all data into 5 subsets, and we used each subset for testing and the other four subsets for training. In the training set, we randomly removed most of the positive instances of the target domain from the training set except for 10% of the labeled data. This gave us the *weakly-supervised* setting. This was repeated five times with different training and test sets. We report the average performance over the five runs.

In our experiments, we set $\mu = 0.8$ in Eq. 1; $\theta = 0.18$ in Eq. 2; and $\gamma = 0.1$ and $\beta = 0.3$ in Eq. 3. For each target domain, we used $k \in \{1,3,5\}$ different source domains chosen randomly from the remaining domains. Thus, the relevance of the source domains to the target domain varies from experiment to experiment.

5.2 Baselines

We compare our framework with several other methods, including state-of-the-art machine learning, relation extraction and common domain adaptation methods. These are described below.

In-domain multiclass classifier This is Support-vector-machine (Fan et al., 2008, SVM) using the one-versus-rest decoding without removing positive labeled data (Jiang and Zhai, 2007b) from the target domain. Its performance can be regarded as an upper bound on the performance of the cross-domain methods.

No-transfer classifier (NT) We only use the few labeled instances of the target relation type together with the negative relation instances to train a binary classifier.

Alternate no-transfer classifier (NT-U) We use the union of the k source-domain labeled data sets D^ss and the small set of target-domain labeled data D_l to train a binary classifier. It is then applied directly to predict on the unlabeled target-domain data D_u without any adaptation.

Laplacian SVM (L-SVM) This is a semi-supervised learning method based on label propagation (Melacci and Belkin, 2011).

Multi-task transfer (MTL) This is a learning method for weakly-supervised relation extraction (Jiang, 2009).

Adaptive domain bootstrapping (DAB) This is an instance-based domain adaptation method for relation extraction (Xu et al., 2010).

Structural correspondence learning (SCL) We use the feature-based domain adaptation approach by Blitzer et al. (2007). We apply SCL on the D^ss and D_l to train a model. The learned model then makes predictions on D_u.

Domain Adaptation Machine (DAM) We use the framework of Duan et al. (2009), which is a multiple-sources domain adaptation method.

For the kernel-based methods above, we use the same composite kernel used in our method. The source codes of L-SVM, MTL, SCL and DAM were obtained from the authors. The others were re-implemented.

5.3 Experimental Results

Tables 2, 3 and 4 present the results on ACE 2004 (corresponding to $k = 1,3,5$), and Tables 5 present those on YAGO (corresponding to $k = 5$).

From Table 3 and Table 5, we see that the proposed method has the best F_1 among all the other methods, except for the supervised upper bound (In-domain). We first notice that NT-U generally does not perform well, and sometimes it performs worse than NT. The reason is that NT-U aims to obtain a consensus among the domains, and this will give a worse label than NT when there are enough irrelevant sources to influence the classification decision wrongly. In fact, one can roughly deduce that a target domain has few relevant source domains by simply comparing columns NT with columns NT-U in the tables: a decrease in F_1 from NT to NT-U suggests that the source domains are mainly irrelevant. For example, for domain BC in ACE 2004, we find that its F_1 decreases from NT to NT-U consistently in Tables 2, 3 and 4, which suggests that BN, NW, CTS, UN and WL are generally irrelevant to it; and similarly for domain CTS. We investigate this further by examining the relevance scores $w_{s,j}$s, and we find that the decreases in F_1 from NT to NT-U happen when there are more regions in the target domain to which source-domains are irrelevant.

We find that MTL, DAB and SCL are better than NT-U when the majority of source domains are relevant. This shows that MTL, DAB and SCL are able to make more effective use of relevant sources than NT-U. Howevever, we find that their perfomances are not stable: for example, MTL for target UN in Table 2. In contrast, we find the performance of L-SVM and DAM to be more stable. The reason is their reduced vulnerability to

Table 2: The F_1 of different methods on ACE 2004 with $k = 1$ source domain. The best performance for each target domain is in bold.

Target	In-domain	NT	NT-U	L-SVM	MTL	DAB	SCL	DAM	RDA
BC	55.74	30.00	20.31	32.42	32.74	32.12	30.41	33.07	**35.43**
BN	67.24	33.43	38.31	35.40	44.81	27.32	45.27	43.26	**47.28**
NW	68.32	41.48	39.35	41.50	42.28	43.27	44.16	41.69	**45.41**
CTS	72.92	36.60	29.90	36.15	**45.06**	37.50	44.68	39.40	44.27
UN	45.16	21.67	17.55	25.10	18.69	18.78	28.77	26.57	**31.07**
WL	46.46	28.53	23.84	29.90	26.13	24.78	23.71	27.01	**30.80**
Average	57.58	31.95	28.21	33.41	35.02	30.46	29.57	33.50	**39.00**

Table 3: The F_1 of different methods on ACE 2004 with $k = 3$ source domains.

Target	In-domain	NT	NT-U	L-SVM	MTL	DAB	SCL	DAM	RDA
BC	55.74	30.00	24.55	32.42	35.26	34.12	37.83	36.08	**39.43**
BN	67.24	33.43	38.31	35.40	49.76	32.15	49.25	45.89	**51.28**
NW	68.32	41.48	43.35	42.50	43.28	43.71	44.16	44.01	**46.41**
CTS	72.92	36.60	30.25	36.15	45.06	37.50	44.68	42.51	**49.27**
UN	45.16	21.67	27.55	25.10	19.72	**35.78**	31.77	33.29	35.07
WL	46.46	28.53	30.72	30.90	33.21	32.81	26.37	32.46	**35.11**
Average	57.58	31.95	32.46	34.20	37.72	36.01	39.01	39.10	**42.76**

Table 4: The F_1 of different methods on ACE 2004 with $k = 5$ source domains.

Target	In-domain	NT	NT-U	L-SVM	MTL	DAB	SCL	DAM	RDA
BC	55.74	30.00	27.32	33.07	37.76	35.08	40.38	38.70	**42.90**
BN	67.24	33.43	40.83	36.42	52.69	42.76.	50.47	48.23	**53.40**
NW	68.32	41.48	44.35	43.69	47.80	44.09	45.50	46.06	**49.13**
CTS	72.92	36.60	34.60	38.90	45.06	38.71	47.35	45.69	**52.63**
UN	45.16	21.67	29.34	26.34	35.47	35.44	33.21	34.13	**36.02**
WL	46.46	28.53	32.41	31.56	34.72	32.81	36.89	32.29	**37.90**
Average	57.58	31.95	34.80	35.0	42.25	38.15	42.30	40.84	**45.33**

negative transfer from irrelevant sources by relying on similarity of feature vectors between source and target domains based on labeled and unlabeled data. Further improvements can still be made, as shown by the better performance of RDA over L-SVM and DAM. This is achieved by further adjusting the relevances between source and target domains according to regions in the target-domain input space.

We analyzed histogram of the relation types to order the domains according to the imbalance of the class distributions. Using this, we observe that MTL, DAB and SCL perform relatively badly when the target-domain distribution is more imbalanced. In constrast, L-SVM, DAM and RDA are more robust.

Comparing with the baselines, RDA achieves the best performance on almost all the experiments. Using the two-phase framework, RDA can successfully transfer useful knowledge even in the pressence of irrelevant sources and imbalanced distributions. For ACE 2004, the improvement in F_1 over the best baseline can be up to 4.0% and is on average 3.6%. Similarly for YAGO, the improvement in F_1 over the best baseline can be up to 5.5% and is on average 4.3%.

Impact of Number of Source Domains Tables 2, 3, 4 and 6 also demonstrate that RDA improves monotonically as the number of source domains increases for both ACE 2004 and YAGO.

Table 5: The F_1 of different methods on YAGO with $k = 5$ source domains.

Target	In-domain	NT	NT-U	L-SVM	MTL	DAB	SCL	DAM	RDA
acquirer_acquiree	58.74	32.12	33.19	43.16	45.28	39.08	44.19	45.07	**51.15**
actedIn	77.36	40.73	44.32	50.45	57.18	49.61	58.23	56.37	**63.40**
bornIn	68.32	42.39	40.35	44.38	49.80	48.36	50.67	48.12	**56.93**
ceo_company	82.92	47.60	51.27	55.27	61.06	58.33	57.41	59.08	**66.71**
company_headquarters	75.16	48.92	52.15	50.13	59.47	61.23	58.36	56.65	**64.36**
created	74.26	46.37	43.58	60.45	60.74	55.08	59.42	57.34	**65.28**
diedIn	81.45	42.78	47.37	57.37	62.69	57.16	65.28	60.44	**71.15**
directed	70.11	44.42	48.29	50.57	54.29	49.09	52.31	50.30	**57.71**
discovered	68.13	37.34	42.51	48.77	53.04	49.82	53.73	51.21	**59.12**
graduatedFrom	69.37	39.28	45.74	51.56	58.22	54.38	56.32	51.17	**60.37**
hasChild	74.56	49.14	50.98	56.07	64.82	53.41	62.38	61.12	**66.83**
hasWonPrize	69.41	38.75	45.72	53.47	57.38	52.76	58.29	54.03	**63.13**
isLeaderOf	79.18	46.31	52.66	58.88	63.49	60.27	63.75	61.51	**70.27**
isMarriedTo	73.33	47.85	48.16	52.31	56.39	50.73	55.35	52.10	**62.58**
livesIn	66.93	36.16	35.15	40.28	50.27	41.72	43.59	48.11	**56.91**
participatedIn	85.38	46.22	48.33	62.48	67.51	61.08	65.38	61.12	**71.72**
person_birthplace	77.62	43.43	45.27	49.66	58.47	59.32	57.55	52.14	**65.80**
person_field	68.32	36.25	37.93	47.69	54.22	50.46	50.47	48.89	**59.47**
politicianOf	79.10	39.17	42.25	53.38	64.56	62.11	60.74	58.82	**68.12**
worksAt	84.29	45.78	49.78	59.34	65.33	65.44	66.53	63.24	**73.31**
Average	74.20	42.55	45.25	52.28	58.21	53.97	56.80	54.84	**63.72**

Performance Gap From Tables 2 to 4, we observe that the smallest performance gap between RDA and the in-domain settings is still high (about 12% with $k = 5$) on ACE 2004. This is because we have used a lot less labeled instances in the target domains: only 10% are used. However, the gaps reduces when the number of source domains increases. Comparing with the in-domain results in Table 5 (which is constant with k), Table 6 also shows a similar trend on YAGO. By exploiting the labeled data in ten source domains in YAGO, our RDA algorithm can reduce the gap between the cross-domain and in-domain settings to 9%.

Table 6: Average F_1 of RDA on YAGO

# source domains	F_1
$k = 1$	53.81
$k = 3$	59.43
$k = 5$	63.72
$k = 10$	65.55

6 Conclusion and Future Work

In this paper, we have proposed a robust domain adaptation (RDA) approach for the relation extraction problem where labeled data is scarce. Existing domain adaptation approaches suffer from negative transfer and under imbalanced distributions. To overcome these, we have proposed a two-phase approach to transfer only relevant information from multiple source domains, and thus derive accurate and robust predictions on the unlabeled target-domain data. Experimental results on ACE 2004 and YAGO have shown that the our domain adaptation method achieves the best performance on F_1 measure compared with the other baselines when only few labeled target instances are used. Because of the practical importance of domain adaptation for relation extraction due to lack of labeled data in new domains, we hope our study and findings will lead to further investigations into this problem.

Acknowledgments

This work is supported by DSO grant DSOCL10021. We thank Jiang for providing the source code for feature extraction and Bollegala for sharing his YAGO dataset.

References

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

Michele Banko, Oren Etzioni, and Turing Center. 2008. The tradeoffs between open and traditional relation extraction. *Proceedings of ACL-08: HLT*, pages 28–36.

Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. 2006. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *The Journal of Machine Learning Research*, 7:2399–2434.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 120–128. ACL.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*.

Danushka Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. 2011. Relation adaptation: learning to extract novel relations with minimum supervision. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*, pages 2205–2210. AAAI Press.

Razvan Bunescu and Raymond Mooney. 2006. Subsequence kernels for relation extraction. *Advances in neural information processing systems*, 18:171–178.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 423–429. ACL.

Hal Daume and D Marcu. 2007. Frustratingly easy domain adaptation. In *Annual meeting-association for computational linguistics*, pages 256–263.

Anhai Doan, Pedro Domingos, and Alon Halevy. 2003. Learning to match the schemas of data sources: A multistrategy approach. *Machine Learning*, 50(3):279–301.

Lixin Duan, Ivor W Tsang, Dong Xu, and Tat-Seng Chua. 2009. Domain adaptation from multiple sources via auxiliary classifiers. In *Proceedings of the 26th Annual ICML*, pages 289–296. ACM.

Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S Weld. 2008. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

Jing Jiang and ChengXiang Zhai. 2007a. Instance weighting for domain adaptation in nlp. In *Annual Meeting-Association For Computational Linguistics*, pages 264–271.

Jing Jiang and ChengXiang Zhai. 2007b. A systematic exploration of the feature space for relation extraction. In *HLT-NAACL*, pages 113–120.

Jing Jiang. 2009. Multi-task transfer learning for weakly-supervised relation extraction. In *Proceedings of the 47th Annual Meeting of the ACL: Volume 2-Volume 2*, pages 1012–1020.

Zornitsa Kozareva and Eduard Hovy. 2010. Not all seeds are equal: Measuring the quality of text mining seeds. In *HLT: The 2010 Annual Conference of the North American Chapter of the ACL*, pages 618–626. ACL.

Stefano Melacci and Mikhail Belkin. 2011. Laplacian support vector machines trained in the primal. *Journal of Machine Learning Research*, 12:1149–1184.

Filipe Mesquita, Jordan Schmidek, and Denilson Barbosa. 2013. Effectiveness and efficiency of open relation extraction. In *Proceedings of EMNLP-13*, volume 500, pages 447–457.

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *Proceedings of NAACL-HLT*, pages 777–782.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th international conference on World wide web*, pages 751–760. ACM.

Marius Pasca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. 2006. Organizing and searching the world wide web of facts - step one: The one-million fact extraction challenge. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, pages 1400–1405. AAAI Press.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1498–1507.

Longhua Qian, Guodong Zhou, Fang Kong, Qiaoming Zhu, and Peide Qian. 2008. Exploiting constituent dependencies for tree kernel-based semantic relation extraction. In *Proceedings of the 22nd Conference on Computational Linguistics*, pages 697–704. ACL.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the National Conference on AI*, pages 474–479.

Michael T Rosenstein, Zvika Marx, Leslie Pack Kaelbling, and Thomas G Dietterich. 2005. To transfer or not to transfer. In *NIPS 2005 Workshop on Transfer Learning*, volume 898, pages –.

Gerard Salton and Michael J. McGill. 1986. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA.

Yusuke Shinyama and Satoshi Sekine. 2006. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the HLT Conference of the North American Chapter of the ACL*, pages 304–311.

Alex J Smola and Bernhard Scholkopf. 1998. *Learning with kernels*. Citeseer.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge unifying wordnet and wikipedia. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM.

Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. 2012. Reducing wrong labels in distant supervision for relation extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 721–729. Association for Computational Linguistics.

Feiyu Xu, Hans Uszkoreit, Sebastian Krause, and Hong Li. 2010. Boosting relation extraction with limited closed-world knowledge. In *Proceedings of the 23rd Conference on Computational Linguistics*, pages 1354–1362. ACL.

Wei Xu, Raphael Hoffmann Le Zhao, and Ralph Grishman. 2013. Filling knowledge base gaps for distant supervision of relation extraction. In *Proceedings of EMNLP-13*, pages 665–670.

Min Zhang, Jie Zhang, Jian Su, and Guodong Zhou. 2006. A composite kernel to extract relations between entities with both flat and structured features. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 825–832. Association for Computational Linguistics.

Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. 2003. Learning with local and global consistency. In *NIPS*.

Jun Zhu, Zaiqing Nie, Xiaojiang Liu, Bo Zhang, and Ji-Rong Wen. 2009. Statsnowball: a statistical approach to extracting entity relationships. In *Proceedings of the 18th international conference on World wide web*, pages 101–110. ACM.

Encoding Relation Requirements for Relation Extraction via Joint Inference

Liwei Chen[1], **Yansong Feng***[1], **Songfang Huang**[2], **Yong Qin**[2] and **Dongyan Zhao**[1]
[1]ICST, Peking University, Beijing, China
[2]IBM China Research Lab, Beijing, China
chenliwei,fengyansong,zhaodongyan@pku.edu.cn
huangsf,qinyong@cn.ibm.com

Abstract

Most existing relation extraction models make predictions for each entity pair locally and individually, while ignoring implicit global clues available in the knowledge base, sometimes leading to conflicts among local predictions from different entity pairs. In this paper, we propose a joint inference framework that utilizes these global clues to resolve disagreements among local predictions. We exploit two kinds of clues to generate constraints which can capture the implicit type and cardinality requirements of a relation. Experimental results on three datasets, in both English and Chinese, show that our framework outperforms the state-of-the-art relation extraction models when such clues are applicable to the datasets. And, we find that the clues learnt automatically from existing knowledge bases perform comparably to those refined by human.

1 Introduction

Identifying predefined kinds of relationship between pairs of entities is crucial for many knowledge base related applications(Suchanek et al., 2013). In the literature, relation extraction (RE) is usually investigated in a classification style, where relations are simply treated as isolated class labels, while their definitions or background information are sometimes ignored. Take the relation *Capital* as an example, we can imagine that this relation will expect a country as its subject and a city as object, and in most cases, a city can be the capital of only one country. All these clues are no doubt helpful, for instance, Yao et al. (2010) explicitly modeled the expected types of a relation's arguments with the help of Freebase's type taxonomy and obtained promising results for RE.

*Yansong Feng is the corresponding author.

However, properly capturing and utilizing such typing clues are not trivial. One of the hurdles here is the lack of off-the-shelf resources and such clues often have to be coded by human experts. Many knowledge bases do not have a well-defined typing system, let alone fine-grained typing taxonomies with corresponding type recognizers, which are crucial to explicitly model the typing requirements for arguments of a relation, but rather expensive and time-consuming to collect. Similarly, the cardinality requirements of arguments, e.g., a person can have only one birthdate and a city can only be labeled as capital of one country, should be considered as a strong indicator to eliminate wrong predictions, but has to be coded manually as well.

On the other hand, most previous relation extractors process each entity pair (we will use *entity pair* and *entity tuple* exchangeably in the rest of the paper) locally and individually, i.e., the extractor makes decisions solely based on the sentences containing the current entity pair and ignores other related pairs, therefore has difficulties to capture possible disagreements among different entity pairs. However, when looking at the output of a multi-class relation predictor globally, we can easily find possible incorrect predictions such as a university locates in two different cities, two different cities have been labeled as capital for one country, a country locates in a city and so on.

In this paper, we will address how to derive and exploit two categories of these clues: the expected types and the cardinality requirements of a relation's arguments, in the scenario of relation extraction. We propose to perform joint inference upon multiple local predictions by leveraging implicit clues that are encoded with relation specific requirements and can be learnt from existing knowledge bases. Specifically, the joint inference framework operates on the output of a sentence level relation extractor as input, derives 5 types of constraints from an existing KB to implicitly capture

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 818–827,
Baltimore, Maryland, USA, June 23-25 2014.

the expected type and cardinality requirements for a relation's arguments, and jointly resolve the disagreements among candidate predictions. We formalize this procedure as a constrained optimization problem, which can be solved by many optimization frameworks. We use integer linear programming (ILP) as the solver and evaluate our framework on English and Chinese datasets. The experimental results show that our framework performs better than the state-of-the-art approaches when such clues are applicable to the datasets. We also show that the automatically learnt clues perform comparably to those refined manually.

In the rest of the paper, we first review related work in Section 2, and in Section 3, we describe our framework in detail. Experimental setup and results are discussed in Section 4. We conclude this paper in Section 5.

2 Related Work

Since traditional supervised relation extraction methods (Soderland et al., 1995; Zhao and Grishman, 2005) require manual annotations and are often domain-specific, nowadays many efforts focus on semi-supervised or unsupervised methods (Banko et al., 2007; Fader et al., 2011). Distant supervision (DS) is a semi-supervised RE framework and has attracted many attentions (Bunescu, 2007; Mintz et al., 2009; Yao et al., 2010; Surdeanu et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012). DS approaches can predict canonicalized (predefined in KBs) relations for large amount of data and do not need much human involvement. Since the automatically generated training datasets in DS often contain noises, there are also research efforts focusing on reducing the noisy labels in the training data (Takamatsu et al., 2012). To bridge the gaps between the relations extracted from open information extraction and the canonicalized relations in KBs, Yao et al. (2012) and Riedel et al. (2013) propose a universal schema which is a union of KB schemas and natural language patterns, making it possible to integrate the unlimited set of uncanonicalized relations in open settings with the relations in existing KBs.

As far as we know, few works have managed to take the relation specific requirements for arguments into account, and most existing works make predictions locally and individually. The MultiR system allows entity tuples to have more than one relations, but still predicts each entity tuple locally (Hoffmann et al., 2011). Surdeanu et al. (2012) propose a two-layer multi-instance multi-label (MIML) framework to capture the dependencies among relations. The first layer is a multi-class classifier making local predictions for single sentences, the output of which are aggregated by the second layer into the entity pair level. Their approach only captures relation dependencies, while we learn implicit relation backgrounds from knowledge bases, including argument type and cardinality requirements. Riedel et al. (2013) propose to use latent vectors to estimate the preferences between relations and entities. These can be considered as the latent type information of the relations' arguments, which is learnt from various data sources. In contrast, our approach learn implicit clues from existing KBs, and jointly optimize local predictions among different entity tuples to capture both relation argument type clues and cardinality clues. Li et al. (2011) and Li et al. (2013) use co-occurring statistics among relations or events to jointly improve information extraction performances in ACE tasks, while we mine existing KBs to collect global clues to solve local conflicts and find the optimal aggregation assignments, regarding existing knowledge facts. de Lacalle and Lapata (2013) encode general domain knowledge as FOL rules in a topic model while our instantiated constraints are directly operated in an ILP model. Zhang et al. (2013) utilize relation cardinality to create negative samples for distant supervision while we use both implicit type clues and relation cardinality expectations to discover possible inconsistencies among local predictions.

3 The Framework

Our framework takes a set of entity pairs and their supporting sentences as its input. We first train a preliminary sentence level extractor which can output confidence scores for its predictions, e.g., a maximum entropy or logistic regression model, and use this local extractor to produce local predictions. In order to implicitly capture the expected type and cardinality requirements for a relation's arguments, we derive two kinds of clues from an existing KB, which are further utilized to discover the disagreements among local candidate predictions. Our objective is to maximize the overall confidence of all the selected predictions.

3.1 Generating Candidate Relations

Since we will focus on the open domain relation extraction, we still follow the distant supervision paradigm to collect our training data guided by a KB, and train the local extractor accordingly. Specifically, we train a sentence level extractor using the maximum entropy model. Given a sentence containing an entity pair, the model will output the confidence of this sentence representing certain relationship (from a predefined relation set) between the entity pair. Formally $\mathcal{R}$ represents the relation set we are working on, $\mathcal{T}$ is the set of entity tuples that we will predict in the test set.

Keep in mind that our local extractor is trained on noisy training data, which, we admit, is not fully reliable. As we observed in a pilot experiment that there is a good chance that the predictions ranked in the second or third may still be correct, we select **top three** predictions as the candidate relations for each mention in order to introduce more potentially correct output.

On the other hand, we should discard the predictions whose confidences are too low to be true, where we set up a threshold of 0.1. For a tuple t, we obtain its candidate relation set by combining the candidate relations of all its mentions, and represent it as R^t. For a candidate relation $r \in R^t$ and a tuple t, we define M_t^r as all t's mentions whose candidate relations contain r. Now the confidence score of a relation $r \in R^t$ being assigned to tuple t can be calculated as:

$$conf(t, r) = \sum_{m \in M_t^r} \text{MEscore}(m, r) \qquad (1)$$

where MEscore(m, r) is the confidence of mention m representing relation r output by our preliminary extractor.

Traditionally, both lexical features and syntactic features are used in relation extraction. Lexical features are the word chains between the subjects and objects in the sentences, while syntactic features are the dependency paths from the subjects to the objects on the dependency graphs of the supporting sentences. However, lexical features are usually too specific to frequently appear in the test data, while the reliability of syntactic features depends heavily on the quality of dependency parsing tools. Generally, we expect more potentially correct relations to be put into the candidate relation set for further consideration. So in addition to lexical and syntactic features, we also use n-gram features to train our preliminary relation extraction model. N-gram features are considered as more ambiguous compared to traditional lexical and syntactic features, and may introduce incorrect predictions, thus improving the recall at the cost of precision.

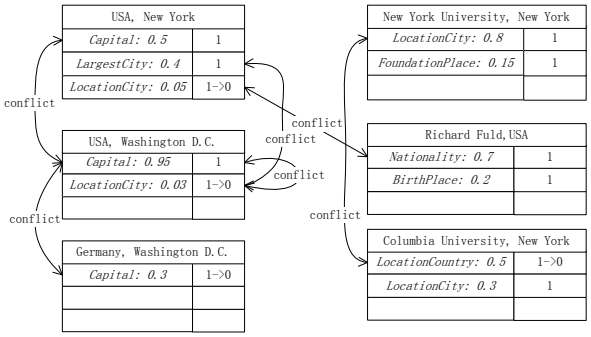


Figure 1: The different types of disagreements we will investigate in the candidate relations. The clues of detecting these inconsistencies can be learnt from a knowledge base.

3.2 Disagreements among the Candidates

The candidate relations we obtained in the previous subsection inevitably include many incorrect predictions. Ideally we should discard those wrong predictions to produce more accurate results.

As discussed earlier, we will exploit from the knowledge base two categories of clues that implicitly capture relations' backgrounds: their expected argument types and argument cardinalities, based on which we can discover two categories of disagreements among the candidate predictions, summarized as argument type inconsistencies and violations of arguments' uniqueness, which have been rarely considered before. We will discuss them in detail, and describe how to learn the clues from a KB afterwards.

Implicit Argument Types Inconsistencies: Generally, the argument types of the correct predictions should be consistent with each other. Given a relation, its arguments sometimes are required to be certain types of entities. For example, in Figure 1, the relation *LargestCity* restricts its subject to be either countries or states, and its object to be cities. If the predictions among different entity tuples require the same entity to belong to different types, we call this

an argument type inconsistency. Take <USA, New York> and <USA, Washington D.C.> as an example. In Figure 1, <USA, New York> has a candidate relation *LargestCity* which restricts *USA* to be either countries or states, while <USA, Washington D.C.> has a prediction *LocationCity* which indicates a disagreement in terms of *USA*'s type because the latter prediction expects *USA* to be an organization located in a city. This warns that at least one of the two candidate relations is incorrect.

The previous scenario shows that the subjects of two candidate relations may disagree with each other. From Figure 1, we can observe two more situations: the first one is that the objects of the two candidate relations are inconsistent with each other, for example <New York University, New York> with the prediction *LocationCity* and <Columbia University, New York> with the prediction *LocationCountry*. The second one is that the subject of one candidate relation do not agree with another prediction's object, for example <Richard Fuld, USA> with the prediction *Nationality* and <USA, New York> with the prediction *LocationCity*. Although we have not assigned explicit types to these entities, we can still exploit the inconsistencies implicitly with the help of shared entities. Note that the implicit argument typing clues here mean whether two relations can share arguments, but NOT enumate what types explicitly their arguments should have.

We formalize all the relation pairs that disagree with each other as follows. These relation pairs can be divided into three subcategories. We represent the relation pairs (r_i, r_j) that are inconsistent in terms of subjects as $\mathcal{C}^{sr}$, the relations pairs that are inconsistent in terms of objects as $\mathcal{C}^{ro}$, the relation pairs that are inconsistent in terms of one's subject and the other one's object as $\mathcal{C}^{rer}$.

It is worth mentioning that disagreements inside a tuple are also included here. For instance, an entity tuple <USA, Washington D.C.> in Figure 1 has two candidate relations, *Capital* and *LocationCity*. These two predictions are inconsistent with each other with respect to the type of *USA*. They implicitly consider *USA* as "country" and "organization", respectively.

Violations of Arguments' Uniqueness: The previous categories of disagreements are all based on the implicit type information of the relations' arguments, Now we make use of the clues of argument cardinality requirements. Given a subject, some relations should have unique objects. For example, in Figure 1, given *USA* as the subject of the relation *Capital*, we can only accept one possible object, because there is great chance that a country only have one capital. On the other hand, given *Washington D.C.* as the object of the relation *Capital*, we can only accept one subject, since usually a city can only be the capital of one country or state. If these are violating in the candidates, we could know that there may be some incorrect predictions. We represent the relations expecting unique objects as $\mathcal{C}^{ou}$, and the relations expecting unique subjects as $\mathcal{C}^{su}$.

3.3 Obtaining the Global Clues

Now, the issue is how to obtain the clues used in the previous subsection. That is, how we determine which relations expect certain types of subjects, which relations expect certain types of objects, etc. These knowledge can be definitely coded by human, or learnt from a KB.

Most existing knowledge bases represent their knowledge facts in the form of (<*subject, relation, object*>) triple, which can be seen as relational facts between entity tuples. Usually the triples in a KB are carefully defined by experts. It is rare to find inconsistencies among the triples in the knowledge base. The clues are therefore learnt from KBs, and further refined manually if needed.

Given two relations r_1 and r_2, we query the KB for all tuples bearing the relation r_1 or r_2. We use S_i and O_i to represent r_i's ($i \in \{1, 2\}$) subject set and object set, respectively. We adopt the pointwise mutual information (PMI) to estimate the dependency between the argument sets of two relations:

$$\text{PMI}(A, B) = \log \frac{p(A, B)}{p(A)p(B)} \qquad (2)$$

where $p(A, B)$ is number of the entities both in A and B, $p(A)$ and $p(B)$ are the numbers of the entities in A and B, respectively. For any pair of relations from $\mathcal{R} \times \mathcal{R}$, we calculate four scores: PMI(S_1, S_2), PMI(O_1, O_2), PMI(S_1, O_2) and PMI(S_2, O_1). To make more stable estimations, we set up a threshold for the PMI. If PMI(S_1, S_2) is lower than the threshold, we will consider that r_1 and r_2 cannot share a subject. Things are similar for the other three scores. The threshold is set to -3 in this paper.

We can also learn the uniqueness of arguments for relations. For each pre-defined relation in $\mathcal{R}$, we collect all the triples containing this relation, and count the portion of the triples which only have one object for each subject, and the portion of the triples which only have one subject for each object. The relations whose portions are higher than the threshold will be considered to have unique argument values. This threshold is set to 0.8 in this paper.

3.4 Integer Linear Program Formulation

As discussed above, given a set of entity pairs and their candidate relations output by a preliminary extractor, our goal is to find an optimal configuration for all those entities pairs jointly, solving the disagreements among those candidate predictions and maximizing the overall confidence of the selected predictions. This is an NP-hard optimization problem. Many optimization models can be used to obtain the approximate solutions.

In this paper, we propose to solve the problem by using an ILP tool, IBM ILOG Cplex[1]. Firstly, for each tuple t and one of its candidate relations r, we define a binary decision variable d_t^r indicating whether the candidate relation r is selected by the solver. Our objective is to maximize the total confidence of all the selected candidates, and the objective function can be written as:

$$\max \sum_{t\in\mathcal{T}, r\in R^t} conf(t,r) d_t^r + \sum_{\forall t, r\in R^t, m\in M_t^r} \max \text{MEscore}(m,r) d_t^r$$

where $conf(t,r)$ is the confidence of the tuple t bearing the candidate relation r. The first component is the sum of the original confidence scores of all the selected candidates, and the second one is the sum of the maximal mention-level confidence scores of all the selected candidates. The latter is designed to encourage the model to select the candidates with higher individual mention-level confidence scores.

We add the constraints with respect to the disagreements described in Section 3.2. For the sake of clarity, we describe the constraints derived from each scenario of the two categories of disagreements separately.

The subject-relation constraints avoid the disagreements between the predictions of two tuples sharing a subject. These constraints can be represented as:

$$d_{t_i}^{r^{t_i}} + d_{t_j}^{r^{t_j}} \leq 1 \quad (3)$$
$$\forall t_i, t_j : subj(t_i) = subj(t_j) \wedge (r^{t_i}, r^{t_j}) \in \mathcal{C}^{sr}$$

where t_i and t_j are two tuples in $\mathcal{T}$, $subj(t_i)$ is the subject of t_i, r^{t_i} is a candidate relation of t_i, r^{t_j} is a candidate relation of t_j.

The object-relation constraints avoid the inconsistencies between the predictions of two tuples sharing an object. Formally we add the following constraints:

$$d_{t_i}^{r^{t_i}} + d_{t_j}^{r^{t_j}} \leq 1 \quad (4)$$
$$\forall t_i, t_j : obj(t_i) = obj(t_j) \wedge (r^{t_i}, r^{t_j}) \in \mathcal{C}^{ro}$$

where $t_i \in \mathcal{T}$ and $t_j \in \mathcal{T}$ are two tuples, $obj(t_i)$ is the object of t_i.

The relation-entity-relation constraints ensure that if an entity works as subject and object in two tuples t_i and t_j respectively, their relations agree with each other. The constraints we add are:

$$d_{t_i}^{r^{t_i}} + d_{t_j}^{r^{t_j}} \leq 1 \quad (5)$$
$$\forall t_i, t_j : obj(t_i) = subj(t_j) \wedge (r^{t_i}, r^{t_j}) \in \mathcal{C}^{rer}$$

The object uniqueness constraints ensure that the relations requiring unique objects do not bear more than one object given a subject.

$$\sum_{t\in Tuple(r), subj(t)=e} d_t^r \leq 1 \quad (6)$$
$$\forall e \wedge r \in \mathcal{C}^{ou}$$

where e is an entity, $Tuple(r)$ are the tuples whose candidate relations contain r.

The subject uniqueness constraints ensure that given an object, the relations expecting unique subjects do not bear more than one subject.

$$\sum_{t\in Tuple(r), obj(t)=e} d_t^r \leq 1 \quad (7)$$
$$\forall e \wedge r \in \mathcal{C}^{su}$$

By adopting ILP, we can combine the local information including MaxEnt confidence scores and the implicit relation backgrounds that are embedded into global consistencies of the entity tuples together. After the optimization problem is solved, we will obtain a list of selected candidate relations for each tuple, which will be our final output.

[1] www.cplex.com

4 Experiments

4.1 Datasets

We evaluate our approach on three datasets, including two English datasets and one Chinese dataset.

The first English dataset, Riedel's dataset, is the one used in (Riedel et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012), with the same split. It uses Freebase as the knowledge base and New York Time corpus as the text corpus, including about 60,000 entity tuples in the training set, and about 90,000 entity tuples in the testing set.

We generate the second English dataset, DBpedia dataset, by mapping the triples in DBpedia (Bizer et al., 2009) to the sentences in New York Time corpus. We map 51 different relations to the corpus and result in about 50,000 entity tuples, 134,000 sentences for training and 30,000 entity tuples, 53,000 sentences for testing.

For the Chinese dataset, we derive knowledge facts and construct a Chinese KB from the Infoboxes of HudongBaike, one of the largest Chinese online encyclopedias. We collect four national economic newspapers in 2009 as our corpus. 28 different relations are mapped to the corpus and this results in 60,000 entity tuples, 120,000 sentences for training and 40,000 tuples, 83,000 sentences for testing.

4.2 Baselines and Competitors

The baseline we use in this paper is Mintz++, which is described in (Surdeanu et al., 2012). It is a modification of the model proposed by Mintz et al. (2009). The model predicts for each mention separately, and allows multi-label outputs for an entity tuple by OR-ing the outputs of its mentions.

As we described in Section 3.1, originally we select the top three predicted relations as the candidates for each mention. In order to investigate whether it is necessary to use up to three candidates, we implement two variants of our approach, which select the top one and top two relations as candidates for each mention, and represented as ILP-1cand and ILP-2cand, respectively.

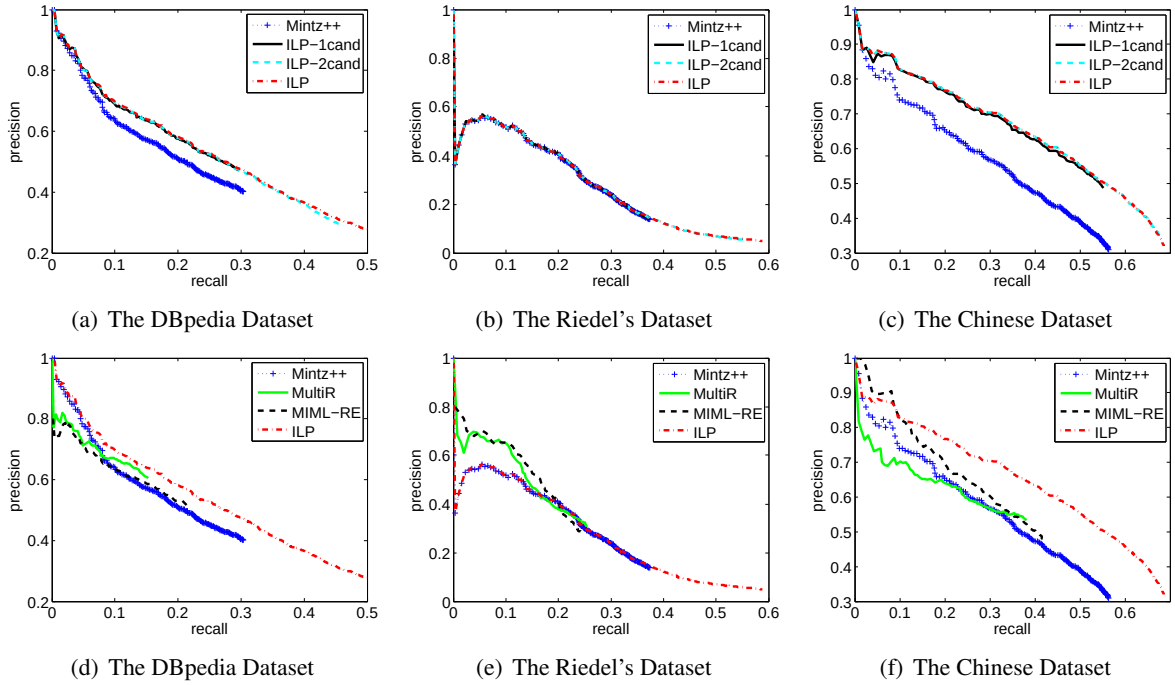
We also use two distant supervision approaches for the comparison. The first one is MultiR (Hoffmann et al., 2011), a novel joint model that can deal with the relation overlap issue. The second one, MIML-RE (Surdeanu et al., 2012), is one of the state-of-the-art MIML relation extraction systems. We tune the models of MultiR and MIML-RE so that they fit our datasets.

4.3 Overall Performance

First we compare our framework and its variants with the baseline and the state-of-the-art RE models. Following previous works, we use the Precision-Recall curve as the evaluation criterion in our experiment. The results are summarized in Figure 2. For the constraints, we first manually select an average of 20 relation pairs for each subcategory of the first kind of clues, and all the relations with unique argument values in $\mathcal{R}$. We also show how automatically learnt clues perform in Section 4.5.

Figure 2 shows that compared with the baseline, our framework performs consistently better in the DBpedia dataset and the Chinese dataset. Mintz++ proves to be a strong baseline on both datasets. It tends to result in a high recall, and its weakness of low precision is perfectly fixed by the ILP model. Our ILP model and its variants all outperform Mintz++ in precision in both datasets, indicating that our approach helps filter out incorrect predictions from the output of MaxEnt model. Compared with MultiR, our framework obtains better results in both datasets. Especially in the Chinese dataset, the improvement in precision reaches as high as 10-16% at the same recall points. Our framework performs better compared to MIML-RE in the English dataset. On the Chinese dataset, our framework outperforms MIML-RE except in the low-recall portion ($<$10%) of the P-R curve. All these results show that embedding the relation background information into RE can help eliminate the wrong predictions and improve the results.

However, in the Riedel's dataset, Mintz++, the MaxEnt relation extractor, does not perform well, and our framework cannot improve its performance. In order to find out the reasons, we manually investigate the dataset. The top three relations of this dataset are */location/location/contains*, */people/person/nationality* and */people/person/place_lived*. About two-thirds of the entity tuples belongs to these three relations, and the outputs of the local extractor usually bias even more to the large relations. What is worse, we cannot find any clues from the top three relations because their arguments' types are too general. Things are similar for many other



(a) The DBpedia Dataset (b) The Riedel's Dataset (c) The Chinese Dataset

(d) The DBpedia Dataset (e) The Riedel's Dataset (f) The Chinese Dataset

Figure 2: Overall performances of our framework and its variants, the baselines and the state-of-the-art approaches on the three datasets.

relations in this dataset. Although we may find some clues any way, they are too few to make any improvement. Hence, our framework does not perform well due to the poor performance of MaxEnt extractor and the lack of clues. To solve this problem, we think of addressing the selection preferences between relations and entities proposed in (Riedel et al., 2013), which should be our future work.

We notice that in all three datasets our variant ILP-1cand is shorter than Mintz++ in recall, indicating we may incorrectly discard some predictions. Compared to ILP-2cand and original ILP, ILP-1cand leads to slightly lower precision but much lower recall, showing that selecting more candidates may help us collect more potentially correct predictions. Comparing ILP-2cand and original ILP, the latter hardly makes any improvement in precision, but is slightly longer in recall, indicating using three candidates can still collect some more potentially correct predictions, although the number may be limited.

In order to study how our framework improves the performances on the DBpedia dataset and the Chinese dataset, we further investigate the number of incorrect predictions eliminated by ILP and the number of incorrect predictions corrected by ILP. We also examine the number of correct predictions newly introduce by ILP, which were NA in Mintz++. We summarize the results in Table 1.

Table 1: Details of the improvements made by ILP in the DBpedia and Chinese datasets.

Datasets	Incorrect Predictions Eliminated	Wrong Predictioins Corrected	Correct Predictions Newly Introduced
DBpedia	268	61	1426
Chinese	1506	14	283

The results show that our framework can reduce the incorrect predictions and introduce more correct predictions at the same time. We also find an interesting results: in the DBpedia dataset, ILP is more likely to introduce correct predictions to the results, while in the Chinese dataset it tends to reduce more incorrect predictions, which may be caused by the differences between performances of Mintz++ on the two datasets, where it gets a higher recall on the Chinese dataset.

Following Surdeanu et al. (2012), we also list the peak F1 score (highest F1 score) for each model in Table 2. Different from (Surdeanu et al., 2012), we use all the entity pairs instead of the ones with more than 10 mentions. We can observe that our model obtains the best performance in the DBpedia dataset and the Chinese dataset. In the DBpedia dataset, it is 3.6% higher than Mintz++,

7.9% higher than MIML-RE and 13.9% higher than MultiR. In the Chinese dataset, Mintz++, MultiR and MIML-RE performs similarly in terms of the highest F1 score, while our model gains about 8% improvement. In the Riedel's dataset, our framework hardly obtains any improvement compared with Mintz++.

We also investigate the impacts of the constraints used in ILP, which are derived based on the two kinds of clues and can encode relation definition information into our framework. Experimental results in Table 2 shows that in the DBpedia dataset, the highest F1 score increases from 35.2% to 38.3% with the help of both kinds of clues, while in the Chinese dataset the improvement is from 44.4% to 52.8%. In the Riedel's dataset we do not see any improvements since there are almost no clues. Furthermore, using constraints derived from only one kind of clues can also improve the performance, but not as well as using both of them.

4.4 Adapting MultiR Sentence Level Extractor to Our Framework

The preliminary relation extractor of our optimization framework is not limited to the MaxEnt extractor, and can take any sentence level relation extractor with confidence scores. We also fit MultiR's mention level extractor into our framework.

As shown in Figure 3, in the DBpedia dataset and the Chinese dataset, in most parts of the curve, ILP optimized MultiR outperforms original MultiR. We think the reason is that our framework make use of global clues to discard the incorrect predictions. The results are not as high as when we use MaxEnt as the preliminary extractor. We think one reason is that MultiR does not perform well in these two datasets. Furthermore, the confidence scores which MultiR outputs are not normalized to the same scale, which brings us difficulties in setting up a confidence threshold to select the candidates. As a result, we only use the top one result as the candidate since including top two predictions without thresholding the confidences performs bad, indicating that a probabilistic sentence-level extractor is more suitable for our framework. We also notice that in the Riedel's dataset our framework does not improve the performance significantly, and we have discussed the reasons in Section 4.3.

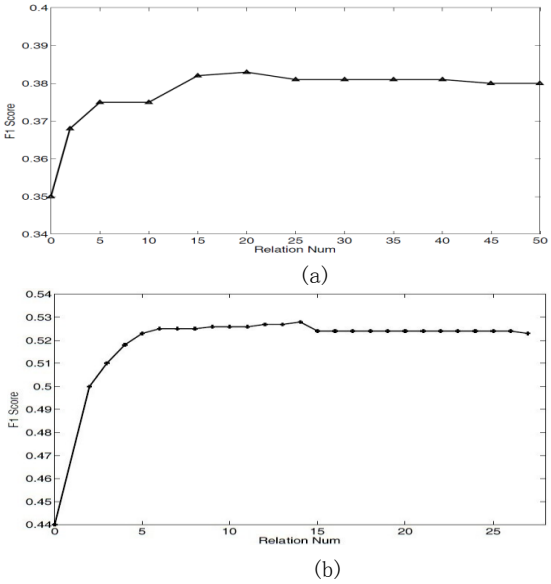
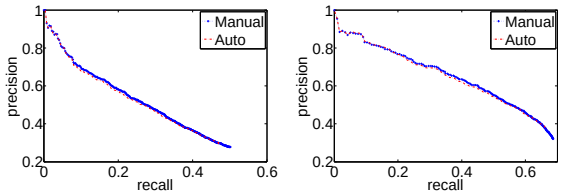


Figure 4: F1 score v.s. number of relations (used to introduce the related learnt clues into the ILP framework) on the DBpedia dataset (a) and the Chinese dataset (b).



(a) The DBpedia Dataset (b) The Chinese Dataset

Figure 5: Performances of manually selected clues and automatically learnt clues on two datasets.

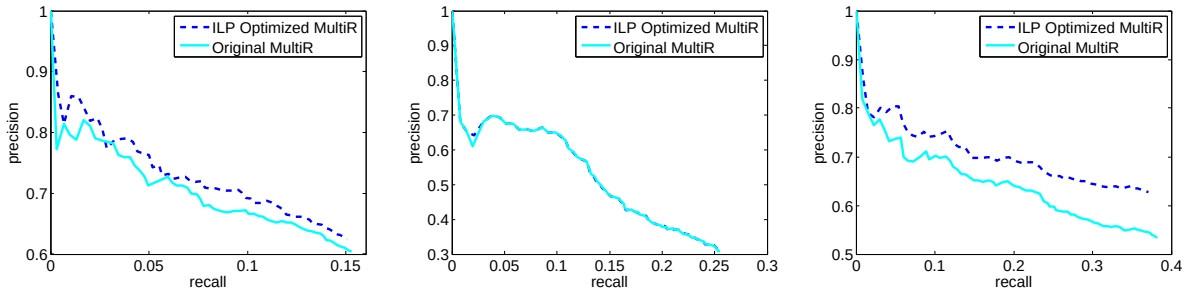
4.5 Examining the Automatically Learnt Clues

Now we evaluate the performance of automatically collected clues used in our model. Since there are almost no clues in the Riedel's dataset, we only investigate the other two datasets. We add clues according to their related relations' proportions in the local predictions. For example, *Country* and *birthPlace* take up about 30% in the local predictions, we thus add clues that are related to these two relations, and then move on with new clues related to other relations according to those relations' proportions in the local predictions.

As is shown in Figure 4, in both datasets, the clues related to more local predictions will solve more inconsistencies, thus are more effective. Adding the first two relations improves the model significantly, and as more relations are added, the

Table 2: Results of the highest F1 score on all three datasets.

	DBpedia			Riedel			Chinese		
Method	**P(%)**	**R(%)**	**F1(%)**	**P(%)**	**R(%)**	**F1(%)**	**P(%)**	**R(%)**	**F1(%)**
Mintz++	40.2	30.5	34.7	35.3	23.2	27.9	43.3	45.7	44.4
MultiR	60.4	15.3	24.4	32.3	25.1	**28.2**	53.5	38.2	44.6
MIML-RE	51.3	21.6	30.4	41.5	19.9	26.9	49.2	41.3	44.9
ILP	37.4	39.2	**38.3**	35.5	23.2	28.0	52.6	52.9	**52.8**
ILP-No-Constraint	34.1	36.3	35.2	35.3	23.2	28.0	43.3	45.7	44.4
ILP-Type-Inconsistent	36.3	39.2	37.7	35.5	23.2	28.0	49.5	49.0	49.2
ILP-Cardinality	35.3	37.8	36.5	35.4	23.2	28.0	50.3	48.8	49.6



(a) The DBpedia Dataset (b) The Riedel's Dataset (c) The Chinese Dataset

Figure 3: The results of original MultiR and ILP optimized MultiR on the three datasets.

performances keep increasing until approaching the still state. It is worth mentioning that when sufficient learnt clues are added into the model, the results are comparable to those based on the clues refined manually, as shown in Figure 5. This indicates that the clues can be collected automatically, and further used to examine whether predicted relations are consistent with the existing ones in the KB, which can be considered as a form of quality control.

5 Conclusions

In this paper, we make use of the global clues derived from KB to help resolve the disagreements among local relation predictions, thus reduce the incorrect predictions and improve the performance of relation extraction. Two kinds of clues, including implicit argument type information and argument cardinality information of relations are investigated. Our framework outperforms the state-of-the-art models if we can find such clues in the KB. Furthermore, our framework is scalable for other local sentence level extractors in addition to the MaxEnt model. Finally, we show that the clues can be learnt automatically from the KB, and lead to comparable performance to manually refined ones.

For future work, we will investigate other kinds of clues and attempt a joint optimization framework that could host entity disambiguation, relation extraction and entity linking together. We will also adopt selection preference between entities and relations since sometimes we may not find useful clues.

Acknowledgments

We would like to thank Heng Ji, Dong Wang and Kun Xu for their useful discussions and the anonymous reviewers for their helpful comments which greatly improved the work. This work was supported by the National High Technology R&D Program of China (Grant No. 2012AA011101), National Natural Science Foundation of China (Grant No. 61272344, 61202233, 61370055) and the joint project with IBM Research.

References

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of IJCAI*, IJCAI'07, pages 2670–2676.

Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak,

and Sebastian Hellmann. 2009. Dbpedia - a crystallization point for the web of data. *Web Semant.*, 7:154–165, September.

Razvan C. Bunescu. 2007. Learning to extract relations from the web using minimal supervision. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL07*.

Oier Lopez de Lacalle and Mirella Lapata. 2013. Unsupervised relation extraction with general domain knowledge. In *EMNLP*, pages 415–425. ACL.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1535–1545, Stroudsburg, PA, USA. Association for Computational Linguistics.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th ACL-HLT - Volume 1*, HLT '11, pages 541–550, Stroudsburg, PA, USA. ACL.

Qi Li, Sam Anzaroot, Wen-Pin Lin, Xiang Li, and Heng Ji. 2011. Joint inference for cross-document information extraction. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 2225–2228, New York, NY, USA. ACM.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *ACL*, pages 73–82. The Association for Computer Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 1003–1011.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases*, volume 6323 of *Lecture Notes in Computer Science*, pages 148–163. Springer Berlin / Heidelberg.

Sebastian Riedel, Limin Yao, Benjamin M. Marlin, and Andrew McCallum. 2013. Relation extraction with matrix factorization and universal schemas. In *Joint Human Language Technology Conference/Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL '13)*, June.

Stephen Soderland, David Fisher, Jonathan Aseltine, and Wendy Lehnert. 1995. Crystal inducing a conceptual dictionary. In *Proceedings of the 14th IJCAI - Volume 2*, IJCAI'95, pages 1314–1319, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Fabian Suchanek, James Fan, Raphael Hoffmann, Sebastian Riedel, and Partha Pratim Talukdar. 2013. Advances in automated knowledge base construction. In *SIGMOD Records journal*, March.

Mihai Surdeanu, David McClosky, Julie Tibshirani, John Bauer, Angel X. Chang, Valentin I. Spitkovsky, and Christopher D. Manning. 2010. A simple distant supervision approach for the TAC-KBP slot filling task. In *Proceedings of the Third Text Analysis Conference (TAC 2010)*, Gaithersburg, Maryland, USA, November.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *EMNLP-CoNLL*, pages 455–465. ACL.

Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. 2012. Reducing wrong labels in distant supervision for relation extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 721–729, Stroudsburg, PA, USA. Association for Computational Linguistics.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2010. Collective cross-document relation extraction without labelled data. In *Proceedings of EMNLP*, EMNLP '10, pages 1013–1023, Stroudsburg, PA, USA. ACL.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2012. Probabilistic databases of universal schema. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, AKBC-WEKEX '12, pages 116–121, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xingxing Zhang, Jianwen Zhang, Junyu Zeng, Jun Yan, Zheng Chen, and Zhifang Sui. 2013. Towards accurate distant supervision for relational facts extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 810–815, Sofia, Bulgaria, August. Association for Computational Linguistics.

Shubin Zhao and Ralph Grishman. 2005. Extracting relations with integrated information using kernel methods. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 419–426, Stroudsburg, PA, USA. Association for Computational Linguistics.

Medical Relation Extraction with Manifold Models

Chang Wang
IBM T. J. Watson Research Center
Yorktown Heights, New York, 10598
changwangnk@gmail.com

James Fan
IBM T. J. Watson Research Center
Yorktown Heights, New York, 10598
fanj@us.ibm.com

Abstract

In this paper, we present a manifold model for medical relation extraction. Our model is built upon a medical corpus containing 80M sentences (11 gigabyte text) and designed to accurately and efficiently detect the key medical relations that can facilitate clinical decision making. Our approach integrates domain specific parsing and typing systems, and can utilize labeled as well as unlabeled examples. To provide users with more flexibility, we also take label weight into consideration. Effectiveness of our model is demonstrated both theoretically with a proof to show that the solution is a closed-form solution and experimentally with positive results in experiments.

1 Introduction

There exists a vast amount of knowledge sources and ontologies in the medical domain. Such information is also growing and changing extremely quickly, making the information difficult for people to read, process and remember. The combination of recent developments in information extraction and the availability of unparalleled medical resources thus offers us the unique opportunity to develop new techniques to help healthcare professionals overcome the cognitive challenges they face in clinical decision making.

Relation extraction plays a key role in information extraction. Using question answering as an example (Wang et al., 2012): in question analysis, the semantic relations between the question focus and each term in the clue can be used to identify the weight of each term so that better search queries can be generated. In candidate answer generation, relations enable the background knowledge base to be used for potential candidate answer generation. In candidate answer scoring, relation-based matching algorithms can go beyond explicit lexical and syntactic information to detect implicit semantic relations shared across the question and passages.

To construct a medical relation extraction system, several challenges have to be addressed:

- The first challenge is how to identify a set of relations that has sufficient coverage in the medical domain. To address this issue, we study a real-world diagnosis related question set and identify a set of relations that has a good coverage of the clinical questions.

- The second challenge is how to efficiently detect relations in a large amount of medical text. The medical corpus underlying our relation extraction system contains 80M sentences (11 gigabytes pure text). To extract relations from a dataset at this scale, the relation detectors have to be fast. In this paper, we speed up relation detectors through parsing adaptation and replacing non-linear classifiers with linear classifiers.

- The third challenge is that the labeled relation examples are often insufficient due to the high labeling cost. When we build a naïve model to detect relations, the model tends to overfit for the labeled data. To address this issue, we develop a manifold model (Belkin et al., 2006) that encourages examples (including both labeled and unlabeled examples) with similar contents to be assigned with similar scores. Our model goes beyond regular regression models in that it applies constraints to those coefficients, such that the topology of the given data manifold will be respected. Computing the optimal weights in a regression model and preserving manifold topology are conflicting objectives, we

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 828–838,
Baltimore, Maryland, USA, June 23-25 2014.

present a closed-form solution to ideally balance these two goals.

The contributions of this paper on medical relation extraction are three-fold:

- The problem setup is new. There is a "fundamental" difference between our problem setup and the conventional setups, like i2b2 (Uzuner et al., 2011). In i2b2 relation extraction task, entity mentions are *manually* labeled, and each mention has 1 of 3 concepts: 'treatment', 'problem', and 'test'. To resemble real-world medical relation extraction challenges where perfect entity mentions do not exist, our new setup requires the entity mentions to be *automatically* detected. The most well-known tool to detect medical entity mentions is MetaMap (Aronson, 2001), which considers all terms as entities and automatically associates each term with a number of concepts from UMLS CUI dictionary (Lindberg et al., 1993) with more than 2.7 million distinct concepts (compared to 3 in i2b2). The huge amount of entity mentions, concepts and noisy concept assignments provide a tough situation that people have to face in real-world applications.
- From the perspective of relation extraction applications, we identify "super relations"- the key relations that can facilitate clinical decision making (Table 1). We also present approaches to collect training data for these relations with a small amount of labeling effort.
- From the perspective of relation extraction methodologies, we present a manifold model for relation extraction utilizing both labeled and unlabeled data. Our approach can also take the label weight into consideration.

The experimental results show that our relation detectors are fast and outperform the state-of-the-art approaches on medical relation extraction by a large margin. We also apply our model to build a new medical relation knowledge base as a complement to the existing knowledge bases.

2 Background

2.1 Medical Ontologies and Sources

Medical domain has a huge amount of natural language content found in textbooks, encyclopedias, guidelines, electronic medical records, and many other sources. It is also growing at an extremely high speed. Substantial understanding of the medical domain has already been included in the Unified Medical Language System (UMLS) (Lindberg et al., 1993), which includes medical concepts, relations, definitions, etc. The 2012 version of the UMLS contains information about more than 2.7 million concepts from over 160 source vocabularies. Softwares for using this knowledge also exist: MetaMap (Aronson, 2001) is able to identify concepts in text. SEMREP (Rindflesch and Fiszman, 2003) can detect some relations using hand-crafted rules.

2.2 Relation Extraction

To extract semantic relations from text, three types of approaches have been applied. Rule-based methods (Miller et al., 2000) employ a number of linguistic rules to capture relation patterns. Feature-based methods (Kambhatla, 2004; Zhao and Grishman, 2005) transform relation instances into a large amount of linguistic features like lexical, syntactic and semantic features, and capture the similarity between these feature vectors. Recent results mainly rely on kernel-based methods. Many of them focus on using tree kernels to learn parse tree structure related features (Collins and Duffy, 2001; Culotta and Sorensen, 2004; Bunescu and Mooney, 2005).

Other researchers study how different approaches can be combined to improve the extraction performance. For example, by combining tree kernels and convolution string kernels, (Zhang et al., 2006) achieved the state of the art performance on ACE data (ACE, 2004). Recently, "distant supervision" has emerged to be a popular choice for training relation extractors without using manually labeled data (Mintz et al., 2009; Jiang, 2009; Chan and Roth, 2010; Wang et al., 2011; Riedel et al., 2010; Ji et al., 2011; Hoffmann et al., 2011; Surdeanu et al., 2012; Takamatsu et al., 2012; Min et al., 2013).

Various relation extraction approaches have been adapted to the medical domain, most of which focus on designing heuristic rules targeted for diagnosis and integrating the medical ontology in the existing extraction approaches. Results of some of these approaches are reported on the i2b2 data (Uzuner et al., 2011).

3 Identifying Key Medical Relations

3.1 Super Relations in Medical Domain

The first step in building a relation extraction system for medical domain is to identify the relations that are important for clinical decision making.

Four main clinical tasks that physicians engage in are discussed in (Demner-Fushman and Lin, 2007). They are **Therapy**- select treatments to offer a patient, taking consideration of effectiveness, risk, cost and other factors (prevention is under the general category of Therapy), **Diagnosis** (including differential diagnosis based on findings and diagnostic test), **Etiology**- identify the factors that cause the disease and **Prognosis**- estimate the patient's likely course over time. These activities can be translated into "search tasks". For example, the search for therapy is usually the therapy selection given a disease.

We did an independent study regarding what clinical questions usually ask for on a set of 5,000 Doctor Dilemma (DD) questions from the American College of Physicians (ACP). This set includes questions about diseases, treatments, lab tests, and general facts[1]. Our analysis shows that about 15% of these questions ask for treatments, preventions or contraindicated drugs for a disease or another way around, 4% are about diagnosis tests, 6% are about the causes of a disease, 1% are about the locations of a disease, 25% are about the symptoms of a disease, 8% are asking for definitions, 7% are about guidelines and the remaining 34% questions either express no relations or some relations that are not very popular.

Based on the analysis in (Demner-Fushman and Lin, 2007) and our own results, we decided to focus on seven key relations in the medical domain, which are described in Table 1. We call these relations "super relations", since they cover most questions in the DD question set and align well with the analysis result in (Demner-Fushman and Lin, 2007).

3.2 Collect Training Data

This section presents how we collect training data for each relation. The overall procedure is illustrated in Figure 1.

[1]Here's an example of these questions and its answer: **Question:** The syndrome characterized by joint pain, abdominal pain, palpable purpura, and a nephritic sediment. **Answer:** Henoch-Schonlein purpura.

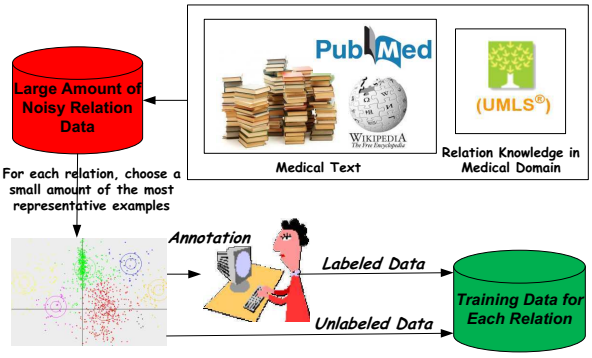


Figure 1: Collect Training Data

Our medical corpus has incorporated a set of medical books/journals[2] and MEDLINE abstracts. We also complemented these sources with Wikipedia articles. In total, the corpus contains 80M sentences (11 gigabyte pure text).

The UMLS 2012 Release contains more than 600 relations and 50M relation instances under around 15 categories. The RO category (RO stands for "has Relationship Other than synonymous, narrower, or broader") is the most interesting one, and covers relations like "may_treat", "has_finding_site", etc. Each relation has a certain number of Concept Unique Identifier (CUI) pairs that are known to bear that relation. In UMLS, some relation information is redundant. Firstly, half of these relations are simply inverse of each other (e.g. the relation "may_treat" and "may_be_treated_by"). Secondly, there is a significant amount of redundancy even among non-inverse relations (e.g. the relation "has_manifestation" and "disease_has_finding").

From UMLS relations, we manually chose a subset of them that are directly related to the super relations discussed in Section 3.1. The correspondences between them are given in Table 1. One thing to note is that super relations are more general than the UMLS relations, and one super relation might integrate multiple UMLS relations. Using the CUI pairs in the UMLS relation knowl-

[2]This is a full list of the books and journals used in our corpus: ACP-Medical Knowledge Self-Assessment Program, EBSCO-Dynamed, EBSCO-Quick Lessons, EBSCO-EBCS, EBSCO-Clinical Review, Wiley-Essential Evidence Plus: EBMG Guidelines, Wiley-Essential Evidence Topics, Wiley-Essential Evidence Plus: EBMG Summaries, Wiley-POEMs, Wiley-The Breast Journal, New England Journal of Medicine, Journal Watch, NCCN-CME, NCCN-GUS, NCCN-Compendium, NCCN-Templates, NCCN-Guidelines for Patients, NCCN-Physician Guidelines, Merck Manual of Diagnosis and Therapy, and UpToDate.

Table 1: Super relations & their arguments, UMLS sources and noise% in the annotation data

Super Relations	Argument 1	Argument 2	UMLS Sources	Noise% in Annotation Data
treats	disease	treatments	may_treat, treats	16%
prevents	disease	treatments	may_prevent	49%
contraindicates	disease	treatments	contraindicated_drug	97%
diagnoses	disease	tests	may_diagnose	63%
causes	disease	causes	cause_of, causative_agent_of	66%
location_of	disease	locations	has_finding_site disease_has_primary_anatomic_site	41%
symptom_of	disease	symptoms	disease_has_finding disease_may_have_finding has_manifestation has_definitional_manifestation	66%

edge base, we associate each super relation with a set of CUI pairs.

To collect the training data for each super relation, we need to collect sentences that express the relation. To achieve this, we parsed all 80M sentences in our medical corpus, looking for the sentences containing the terms that are associated with the CUI pairs in the knowledge base. This (distant supervision) approach resulted in a huge amount of sentences that contain the desired relations, but also brought in a lot of noise in the form of false positives. For example, we know from the knowledge base that "antibiotic drug" may treat "Lyme disease". However the sentence "This paper studies the relationship between antibiotic drug and Lyme disease" contains both terms but does not express the "treats" relation.

The most reliable way to clean the training data is to ask annotators to go through the sentences and assign the sentences with positive/negative labels. However, it will not work well when we have millions of sentences to vet. To minimize the human labeling effort, we ran a K-medoids clustering on the sentences associated with each super relation and kept the cluster centers as the most representative sentences for annotation. Depending on the number of the sentences we collected for each relation, the #clusters was chosen from 3,000 - 6,000. The similarity of two sentences is defined as the bag-of-words similarity of the dependency paths connecting arguments. Part of the resulting data was manually vetted by our annotators, and the remaining was held as unlabeled data for further experiments.

Our relation annotation task is quite straightforward, since both arguments are given and the decision is a Yes-or-No decision. The noise rate of each relation (#sentences expressing the relation / #sentences) is reported in Table 1 based on the annotation results. The noise rates differ significantly from one relation to another. For "treats" relation, only 16% of the sentences are false positives. For "contraindicates" relation, the noise rate is 97%.

To grow the size of the negative training set for each super relation, we also added a small amount of the most representative examples (also coming from K-medoids clustering) from each unrelated UMLS relation to the training set as negative examples. This resulted in more than 10,000 extra negative examples for each relation.

3.3 Parsing and Typing

The most well-known tool to detect medical entity mentions is MetaMap (Aronson, 2001), which considers all terms as entities and automatically associates each term with a number of concepts from UMLS CUI dictionary (Lindberg et al., 1993) with 2.7 million distinct concepts.

The parser used in our system is MedicalESG, an adaptation of ESG (English Slot Grammar) (McCord et al., 2012) to the medical domain with extensions of medical lexicons integrated in the UMLS 2012 Release. Compared to MetaMap, MedicalESG is based on the same medical lexicons, 10 times faster and produces very similar parsing results.

We use the semantic types defined in UMLS (Lindberg et al., 1993) to categorize argument types. The UMLS consists of a set of 133 subject categories, or semantic types, that provide a consistent categorization of more than 2M concepts represented in the UMLS Metathesaurus. Our system assigns each relation argument with one or more UMLS semantic types through a two step process. Firstly, we use MedicalESG to process the input sentence, identify segments of text that correspond to concepts in

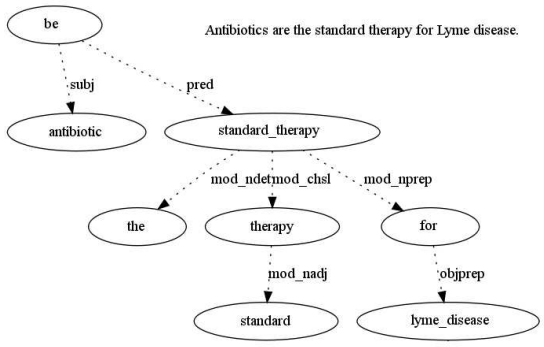


Figure 2: A Parse Tree Example

the UMLS Metathesaurus and associate each of them with one or more UMLS CUIs (Concept Unique Identifier). Then we do a CUI lookup in UMLS to find the corresponding semantic types for each CUI.

Most relation arguments are associated with multiple semantic types. For example, the term "tetracycline hydrochloride" has two types: "Organic Chemical" and "Antibiotic". Sometimes, the semantic types are noisy due to ambiguity of terms. For example, the term "Hepatitis b" is associated with both "Pharmacologic Substance" and "Disease or Syndrome" based on UMLS. The reason for this is that people use "Hepatitis b" to represent both "the disease of Hepatitis b" and "Hepatitis b vaccine", so UMLS assigns both types to it. This is a concern for relation extraction, since two types bear opposite meanings. Our current strategy is to integrate all associated types, and rely on the relation detector trained with the labeled data to decide how to weight different types based upon the context.

Here is an illustrative example. Consider the sentence: *"Antibiotics are the standard therapy for Lyme disease"*: MedicalESG first generates a dependency parse tree (Figure 2) to represent grammatical relations between the words in the sentence, and then associates the words with CUIs. For example, "Antibiotics" is associated with CUI "C0003232" and "Lyme disease" is associated with two CUIs: "C0024198" and "C0717360". CUI lookup will assign "Antibiotics" with a semantic type "Antibiotic", and "Lyme disease" with three semantic types: "Disease or Syndrome", "Pharmacologic Substance" and "Immunologic Factor". This sentence expresses a "treats" relation between "Antibiotics" and "Lyme disease".

4 Relation Extraction with Manifold Models

4.1 Motivations

Given a few labeled examples and many unlabeled examples for a relation, we want to build a relation detector leveraging both labeled and unlabeled data. Following the manifold regularization idea (Belkin et al., 2006), our strategy is to learn a function that assigns a score to each example. Scores are fit so that examples (both labeled and unlabeled) with similar content get similar scores, and scores of labeled examples are close to their labels. Integration of the unlabeled data can help solve overfitting problems when the labeled data is not sufficient.

4.2 Features

We use 8 groups of features to represent each relation example. These features are commonly used for relation extraction.

- (1) Semantic types of argument 1, such as "Antibiotic".
- (2) Semantic types of argument 2.
- (3) Syntactic features representing the dependency path between two arguments, such as "subj", "pred", "mod_nprep" and "objprep" (between arguments "antibiotic" and "lyme disease") in Figure 2.
- (4) Features modeling the incoming and outgoing links of both arguments. These features are useful to determine if a relation goes from argument 1 to argument 2 or vice versa.
- (5) Topic features modeling the words in the dependency path. In the example given in Figure 2, the dependency path contains the following words: "be", "standard therapy" and "for". These features as well as the features in (6) are achieved by projecting the words onto a 100 dimensional LSI topic space (Deerwester et al., 1990) constructed from our medical corpus.
- (6) Topic features modeling the words in the whole sentence.
- (7) Bag-of-words features modeling the dependency path. In (7) and (8), we only consider the words that have occurred in the positive training data.

Notations:

The input dataset $X = \{x_1, \cdots, x_m\}$ is represented as a feature-instance matrix.

The desired label vector $Y = \{y_1, \cdots, y_l\}$ represents the labels of $\{x_1, \cdots, x_l\}$, where $l \leq m$.

W is a weight matrix, where $W_{i,j} = e^{-\|x_i - x_j\|^2}$ models the similarity of x_i and x_j.

$\|x_i - x_j\|$ stands for the Euclidean distance between x_i and x_j in the vector space.

D is a diagonal matrix: $D_{i,i} = \sum_j W_{i,j}$.

$\mathcal{L} = D^{-0.5}(D - W)D^{-0.5}$ is called normalized graph Laplacian matrix.

Δ is a user defined $l \times l$ diagonal matrix, where Δ_i represents the weight of label y_i.

$\mathcal{A} = \begin{pmatrix} \Delta & 0 \\ 0 & 0 \end{pmatrix}$ is an $m \times m$ matrix.

$V = [y_1, \cdots y_l, 0, \cdots, 0]$ is a $1 \times m$ matrix.

μ is a weight scalar.

$()^+$ represents pseudo inverse.

Algorithm:

1. **Represent each example using features: $X = \{x_1, \cdots, x_m\}$, where x_i is the ith example.**
2. **Construct graph Laplacian matrix $\mathcal{L}$ modeling the data manifold.**
3. **Construct vector $V = [y_1, \cdots y_l, 0, \cdots, 0]$.**
4. **Compute projection function f for each relation: $f = (X(\mathcal{A} + \mu\mathcal{L})X^T)^+ X\mathcal{A}V^T$.**

Figure 3: Notations and the Algorithm to Train a Manifold Model for Relation Extraction

- (8) Bag-of-words features modeling the whole sentence.

In relation extraction, many recent approaches use non-linear kernels to get the similarity of two relation examples. To classify a relation example, a lot of dot product computations are required. This is very time consuming and becomes a bottleneck in using relation extraction to facilitate clinical decision making. To speed up the classifier during the apply time, we decided to use a linear classifier instead of non-linear classifiers.

We represent all features in a single feature space. For example, we use a vector of 133 entries (UMLS contains 133 semantic types) to represent the types of argument 1. If argument 1 is associated with two types: "Organic Chemical" and "Antibiotic", we set the two corresponding entries to 1 and all the other entries to 0. Similar approaches are used to represent the other features.

4.3 The Main Algorithm

The problem we want to solve is formalized as follows: given a relation dataset $X = \{x_1, \cdots, x_m\}$, and the desired label $Y = \{y_1, \cdots, y_l\}$ for $\{x_1, \cdots, x_l\}$, where $l \leq m$, we want to construct a mapping function f to project any example x_i to a new space, where $f^T x_i$ matches x_i's desired label y_i. In addition, we also want f to preserve the manifold topology of the dataset, such that similar examples (both labeled and unlabeled) get similar scores. Here, the label is '+1' for positive examples, and '-1' for negative examples. Notations and the main algorithm to construct f for each relation are given in Figure 3.

4.4 Justification

The solution to the problem defined in Section 4.3 is given by the mapping function f to minimize the following cost function:

$$C(f) = \sum_{i \leq l} \alpha_i (f^T x_i - y_i)^2 + \mu \sum_{i,j} W_{i,j} (f^T x_i - f^T x_j)^2.$$

The first term of $C(f)$ is based on labeled examples, and penalizes the difference between the mapping result of x_i and its desired label y_i. α_i is a user specified parameter, representing the weight of label y_i. The second term of $C(f)$ does not take label information into account. It encourages the neighborhood relationship (geometry of the manifold) within X to be preserved in the mapping. When x_i and x_j are similar, the corresponding $W_{i,j}$ is big. If f maps x_i and x_j to different positions, f will be penalized. The second term is useful to bound the mapping function f and prevents overfitting from happening. Here μ is the weight of the second term. When $\mu = 0$, the model disregards the unlabeled data, and the data manifold topology is not respected.

Compared to manifold regularization (Belkin et al., 2006), we do not include the RKHS norm term. Instead, we associate each labeled example with an extra weight for label confidence. This weight is particularly useful when the training data comes from "Crowdsourcing", where we ask

multiple workers to complete the same task to correct errors. In that scenario, weights can be assigned to labels based upon annotator agreement.

Theorem 1: $f = (X(\mathcal{A} + \mu\mathcal{L})X^T)^+ X\mathcal{A}V^T$ minimizes the cost function $C(f)$.
Proof:
Given the input X, we want to find the optimal mapping function f such that $C(f)$ is minimized:

$$f = \arg\min_f C(f).$$

It can be verified that

$$\sum_{i \leq l} \alpha_i (f^T x_i - y_i)^2 = f^T X\mathcal{A}X^T f - 2f^T X\mathcal{A}V^T + V\mathcal{A}V^T.$$

We can also verify that

$$\mu \sum_{i,j} (f^T x_i - f^T x_j)^2 W_{i,j} = \mu f^T X\mathcal{L}X^T f.$$

So $C(f)$ can be written as

$$f^T X\mathcal{A}X^T f - 2f^T X\mathcal{A}V^T + V\mathcal{A}V^T + \mu f^T X\mathcal{L}X^T f.$$

Using the Lagrange multiplier trick to differentiate $C(f)$ with respect to f, we have

$$2X\mathcal{A}X^T f + 2\mu X\mathcal{L}X^T f = 2X\mathcal{A}V^T.$$

This implies that

$$X(\mathcal{A} + \mu\mathcal{L})X^T f = X\mathcal{A}V^T.$$

So

$$f = (X(\mathcal{A} + \mu\mathcal{L})X^T)^+ X\mathcal{A}V^T,$$

where "+" represents pseudo inverse. ∎

4.5 Advantages

Our algorithm offers the following advantages:

- The algorithm exploits unlabeled data, which helps prevent "overfitting" from happening.
- The algorithm provides users with the flexibility to assign different labels with different weights. This feature is useful when the training data comes from "crowdsourcing" or "distant supervision".
- Different from many approaches in this area, our algorithm provides a closed-form solution of the result. The solution is global optimal regarding the cost function $C(f)$.
- The algorithm is computationally efficient at the apply time (as fast as linear regressions).

5 Experiments

5.1 Cross-Validation Test

We use a cross-validation test[3] with the relation data generated in Section 3.2 to compare our approaches against the state-of-the-art approaches. The task is to classify the examples into positive or negative for each relation. We applied a 5-fold cross-validation. In each round of validation, we used 20% of the data for training and 80% for testing. The F_1 scores reported here are the average of all 5 rounds. We used MedicalESG to process the input text for all approaches.

5.1.1 Data and Parameters

This dataset includes 7 relations. We do not consider the relation of "contraindicates" in this test, since it has too few positive examples. On average, each relation contains about 800 positive examples and more than 13,000 negative examples. To eliminate the examples that are trivial to classify, we removed the negative examples that do not bear the valid argument types. This removed the examples that can be easily classified by a type filter, resulting in 3,000 negatives on average per relation. For each relation, we also collected 5,000 unlabeled examples and put them into two sets: unlabeled set 1 and 2 (2,500 examples in each set).

No parameter tuning was taken and no relation specific heuristic rules were applied in all tests. In all manifold models, $\mu = 1$. In SVM implementations, the trade-off parameter between training error and margin was set to 1 for all experiments.

5.1.2 Baseline Approaches

We compare our approaches to three state-of-the-art approaches including SVM with convolution tree kernels (Collins and Duffy, 2001), linear regression and SVM with linear kernels (Schölkopf and Smola, 2002). To adapt the tree kernel to medical domain, we followed the approach in (Nguyen et al., 2009) to take the syntactic structures into consideration. We also added the argument types as features to the tree kernel. In the tree kernel implementation, we assigned the tree structure and the vector corresponding to the argument types

[3] If we take the perfect entity mentions and the associated concepts provided by i2b2 (Uzuner et al., 2011) as the input, our system can directly apply to i2b2 relation extraction data. However, the i2b2 data has a tough data use agreement. Our legal team held several rounds of negotiations with the i2b2 data owner and then decided we should not use it due to the high legal risks. We are not aware of other available medical relation extraction datasets that fit for our evaluations.

Table 2: F_1 Scores from a Five-Fold Cross Validation Experiment

	SVM Tree Kernel	SVM Linear Kernel	Linear Regression	Manifold *Unlabeled*	Manifold *Predicted Labels with Weights*	Manifold *Predicted Labels without Weights*	Manifold *Unlabeled+Predicted Labels with Weights*
treats	0.7648	0.7850	0.7267	0.8025	0.8041	0.7884	**0.8085**
prevents	0.2859	0.3887	0.3922	0.5502	0.5696	**0.6349**	0.6332
causes	0.3885	0.5024	0.5219	**0.5779**	0.5088	0.3978	0.5081
location_of	0.6113	0.6009	0.4968	0.7275	0.7363	0.6964	**0.7454**
diagnoses	0.5520	0.4934	0.3202	0.6468	0.6485	0.5720	**0.6954**
symptom_of	0.4398	0.5611	0.5984	**0.6347**	0.5314	0.4515	0.5968
average	0.5071	0.5553	0.5094	0.6566	0.6331	0.5902	**0.6646**

with equal weights. The SVM with linear kernels and the linear regression model used the same features as the manifold models.

5.1.3 Settings for the Manifold Models

We tested our manifold model for each relation under three different settings:

(1) Manifold *Unlabeled*: We combined the labeled data and unlabeled set 1 in training. We set $\alpha_i = 1$ for $i \in [1, l]$.

(2) Manifold *Predicted Labels*: We combined labeled data and unlabeled set 2 in training. $\alpha_i = 1$ for $i \in [1, l]$. Different from the previous setting, we gave a label estimation to all the examples in the unlabeled set 2 based on the noise rate (Noise%) from Table 1. The label of all unlabeled examples was set to "+1" when $100\% - 2 \cdot Noise\% > 0$, or "-1" otherwise. Two weighting strategies were applied:

- *With Weights:* We let label weight $\alpha_i = |100\% - 2 \cdot Noise\%|$ for all x_i coming from the unlabeled set 2. This setting represents an empirical rule to estimate the label and confidence of each unlabeled example based on the sampling result.
- *Without Weights:* α_i is always set to 1.

(3) Manifold *UnLabeled+Predicted Labels*: a combination of setting (1) and (2). In this setting, the data from unlabeled set 1 was used as unlabeled data and the data from unlabeled set 2 was used as labeled data (With Weights).

5.1.4 Results

The results are summarized in Table 2.

The tree kernel-based approach and linear regression achieved similar F_1 scores, while linear SVM made a 5% improvement over them. One thing to note is that the results from these approaches vary significantly. The reason for this is that the labeled training data is not sufficient. So the approaches that completely depend on the labeled data are likely to run into overfitting. Linear SVM performed better than the other two, since the large-margin constraint together with the linear model constraint can alleviate overfitting.

By integrating unlabeled data, the manifold model under setting (1) made a 15% improvement over linear regression model on F_1 score, where the improvement was significant across all relations.

Under setting (2), the *With Weights* strategy achieved a slightly worse F_1 score than the previous setting but much better result than the baseline approaches. This tells us that estimating the label of unlabeled examples based upon the sampling result is one way to utilize unlabeled data and may help improve the relation extraction results. The results also show that the label weight is important for this setting, since the *Without Weights* strategy did not perform very well.

Compared to setting (1) and (2), setting (3) made use of 2,500 more unlabeled examples, and achieved the best performance among all approaches. On one hand, this result shows that using more unlabeled data can further improve the result. On the other hand, the insignificant improvement over (1) and (2) strongly indicates that how to utilize more unlabeled data to achieve a significant improvement is non-trivial and deserves more attention. To what extensions the unlabeled data can help the learning process is an open problem. Generally speaking, when the existing data is sufficient to characterize the dataset geometry, adding more unlabeled data will not help (Singh et al., 2008).

We tested the tree kernel-based approach without integrating the medical types as well. That resulted in very poor performance: the average F_1 score was below 30%. We also applied the rules used in SEMREP (Rindflesch and Fiszman, 2003) to this dataset. Since the relations detected by

SEMREP rules cannot be perfectly aligned with super relations, we cannot directly compare the results. Overall speaking, SEMREP rules are very conservative and detect very few relations from the same text.

5.2 Knowledge Base (KB) Construction

The UMLS Metathesaurus (Lindberg et al., 1993) contains a large amount of manually extracted relation knowledge. Such knowledge is invaluable for people to collect training data to build new relation detectors. One downside of using this KB is its incompleteness. For example, it only contains the treatments for about 8,000 diseases, which are far from sufficient. Further, the medical knowledge is changing extremely quickly, making people hard to understand it, and update it in the knowledge base in a timely manner.

To address these challenges, we constructed our own relation KB as a complement to the UMLS relation KB. We directly ran our relation detectors (trained with all labeled and unlabeled examples) on our medical corpus to extract relations. Then we combined the results and put them in a new KB. The new KB covers all super relations and stores the knowledge in the format of (relation_name, argument_1, argument_2, confidence), where the confidence is computed based on the relation detector confidence score and relation popularity in the corpus. The most recent version of our relation KB contains 3.4 million such entries.

We compared this new KB against UMLS KB using an answer generation task on a set of 742 Doctor Dilemma questions. We first ran our relation detectors to detect the relation(s) in the question clue involving question focus (what the question asks for). Then we searched against both KBs using the relation name and the non-focus argument for the missing argument. The search results were then generated as potential answers. We used the same relations to do KB lookup, so the results are directly comparable. Since most questions only have one correct answer, the precision number is not very important in this experiment.

If we detect multiple relations in the question, and the same answer is generated from more than one relations, we sum up all those confidence scores to make such answers more preferable. Sometimes, we may generate too many answers from KBs. For example, if the detected relation is "location_of" and the non-focus argument is "skin", then thousands of answers can be generated. In this scenario, we sort the answers based upon the confidence scores and only consider up to p answers for each question. In our test, we considered three numbers for p: 20, 50 and 3,000.

From Table 3, we can see that the new KB outperforms the most popularly-used UMLS KB at all recall levels by a large margin. This result indicates that the new KB has a much better knowledge coverage. The UMLS KB is manually created and thus more precise. In our experiment, the UMLS KB generated fewer answers than the new KB. For example, when up to 20 answers were generated for each question, the UMLS KB generated around 4,700 answers for the whole question set, while the new KB generated about 7,600 answers.

Construction of the new KB cost 16 machines (using 4×2.8G cores per machine) 8 hours. The reported computation time is for the whole corpus with 11G pure text.

Table 3: Knowledge Base Comparison

	Recall@20	Recall@50	Recall@3000
Our KB	135/742	182/742	301/742
UMLS KB	42/742	52/742	73/742

6 Conclusions

In this paper, we identify a list of key relations that can facilitate clinical decision making. We also present a new manifold model to efficiently extract these relations from text. Our model is developed to utilize both labeled and unlabeled examples. It further provides users with the flexibility to take label weight into consideration. Effectiveness of the new model is demonstrated both theoretically and experimentally. We apply the new model to construct a relation knowledge base (KB), and use it as a complement to the existing manually created KBs.

Acknowledgments

We thank Siddharth Patwardhan for help on tree kernels, Sugato Bagchi and Dr. Herbert Chase's team for categorizing the Doctor Dilemma questions. We also thank Anthony Levas, Karen Ingraffea, Mark Mergen, Katherine Modzelewski, Jonathan Hodax, Matthew Schoenfeld and Adarsh Thaker for vetting the training data.

References

ACE. 2004. The automatic content extraction projects, http://projects.ldc.upenn.edu/ace/.

A. Aronson. 2001. Effective mapping of biomedical text to the UMLS metathesaurus: the MetaMap program. In *Proceedings of the 2001 Annual Symposium of the American Medical Informatics Association*.

M. Belkin, P. Niyogi, and V. Sindhwani. 2006. Manifold regularization: a geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, pages 2399–2434.

R. Bunescu and R. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*.

Y. Chan and D. Roth. 2010. Exploiting background knowledge for relation extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 152–160.

M. Collins and N. Duffy. 2001. Convolution kernels for natural language. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, pages 625–632.

A. Culotta and J. Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 423–429.

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

D. Demner-Fushman and J. Lin. 2007. Answering clinical questions with knowledge-based and statistical techniques. *Journal of Computational Linguistics*, 56:63–103.

R. Hoffmann, C. Zhang, X. Ling, L. Zettlemoyer, and D. S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

H. Ji, R. Grishman, and H. T. Dang. 2011. Overview of the TAC 2011 knowledge base population track. In *Proceedings of the Text Analytics Conference*.

J. Jiang. 2009. Multi-task transfer learning for weakly-supervised relation extraction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics (ACL) and the 4th International Joint Conference on Natural Language Processing (IJCNLP)*, pages 1012–1020.

N. Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*.

D. Lindberg, B. Humphreys, and A. McCray. 1993. The Unified Medical Language System. *Methods of Information in Medicine*, 32:281–291.

M. McCord, J. W. Murdock, and B. K. Boguraev. 2012. Deep parsing in Watson. *IBM Journal of Research and Development*, 56.

S. Miller, H. Fox, L. Ramshaw, and R. Weischedel. 2000. A novel use of statistical parsing to extract information from text. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*.

B. Min, R. Grishman, L. Wan, C. Wang, and D. Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *The 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

M. Mintz, S. Bills, R. Snow, and D. Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics (ACL) and the 4th International Joint Conference on Natural Language Processing (IJCNLP)*, pages 1003–1011.

T. Nguyen, A. Moschitti, and G. Riccardi. 2009. Convolution kernels on constituent, dependency and sequential structures for relation extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

S. Riedel, L. Yao, and A. McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*.

T. C. Rindflesch and M. Fiszman. 2003. The interaction of domain knowledge and linguistic structure in natural language processing: interpreting hypernymic propositions in biomedical text. *Journal of Biomedical Informatics*, 36:462–477.

B. Schölkopf and A. J. Smola. 2002. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press.

A. Singh, R. D. Nowak, and X. Zhu. 2008. Unlabeled data: now it helps, now it doesnot. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*.

M. Surdeanu, J. Tibshirani, R. Nallapati, and C. D. Manning. 2012. Multi-instance multilabel learning for relation extraction. In *Proceedings of the 2012*

Conference on Empirical Methods in Natural Language Processing and Natural Language Learning (EMNLP).

S. Takamatsu, I. Sato, and H. Nakagawa. 2012. Reducing wrong labels in distant supervision for relation extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ö. Uzuner, B. R. South, S. Shen, and S. L. DuVall. 2011. 2010 i2b2/VA challenge on concepts, assertions, and relations in clinical text. *Journal of American Medical Informatics Association*, 18:552–556.

C. Wang, J. Fan, A. Kalyanpur, and D. Gondek. 2011. Relation extraction with relation topics. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

C. Wang, A. Kalyanpur, J. Fan, B. Boguraev, and D. Gondek. 2012. Relation extraction and scoring in DeepQA. *IBM Journal of Research and Development*, 56.

M. Zhang, J. Zhang, J. Su, and G. Zhou. 2006. A composite kernel to extract relations between entities with both flat and structured features. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (ACL)*.

S. Zhao and R. Grishman. 2005. Extracting relations with integrated information using kernel methods. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 419–426.

Distant Supervision for Relation Extraction with Matrix Completion

Miao Fan[†,‡,*], **Deli Zhao**[‡], **Qiang Zhou**[†], **Zhiyuan Liu**[◇,‡], **Thomas Fang Zheng**[†], **Edward Y. Chang**[‡]
[†] CSLT, Division of Technical Innovation and Development,
Tsinghua National Laboratory for Information Science and Technology, Tsinghua University, China.
[◇] Department of Computer Science and Technology, Tsinghua University, China.
[‡] HTC Beijing Advanced Technology and Research Center, China.
[*]fanmiao.cslt.thu@gmail.com

Abstract

The essence of distantly supervised relation extraction is that it is an *incomplete* multi-label classification problem with *sparse* and *noisy* features. To tackle the sparsity and noise challenges, we propose solving the classification problem using matrix completion on factorized matrix of minimized rank. We formulate relation classification as completing the unknown labels of testing items (entity pairs) in a sparse matrix that concatenates training and testing textual features with training labels. Our algorithmic framework is based on the assumption that the rank of item-by-feature and item-by-label joint matrix is low. We apply two optimization models to recover the underlying low-rank matrix leveraging the sparsity of feature-label matrix. The matrix completion problem is then solved by the fixed point continuation (FPC) algorithm, which can find the global optimum. Experiments on two widely used datasets with different dimensions of textual features demonstrate that our low-rank matrix completion approach significantly outperforms the baseline and the state-of-the-art methods.

1 Introduction

Relation Extraction (RE) is the process of generating structured relation knowledge from unstructured natural language texts. Traditional supervised methods (Zhou et al., 2005; Bach and Badaskar, 2007) on small hand-labeled corpora, such as MUC[1] and ACE[2], can achieve high precision and recall. However, as producing hand-labeled corpora is laborius and expensive, the supervised approach can not satisfy the increasing demand of building large-scale knowledge repositories with the explosion of Web texts. To address the lacking training data issue, we consider the distant (Mintz et al., 2009) or weak (Hoffmann et al., 2011) supervision paradigm attractive, and we improve the effectiveness of the paradigm in this paper.

Entity pair	<Barack Obama, U.S.>
Relation instances from knowledge bases	1. President of (Barack Obama, U.S.) 2. Born in (Barack Obama, U.S.)
Relation mentions from free texts	1. **Barack Obama** is the 44th and current President of the **U.S.**. (President of) 2. **Barack Obama** ended **U.S.** military involvement in the Iraq War. (-) 3. **Barack Obama** was born in Honolulu, Hawaii, **U.S.**. (Born in) 4. **Barack Obama** ran for the **U.S.** Senate in 2004. (Senate of)

Figure 1: Training corpus generated by the basic alignment assumption of distantly supervised relation extraction. The relation instances are the triples related to President Barack Obama in the Freebase, and the relation mentions are some sentences describing him in the Wikipedia.

The intuition of the paradigm is that one can take advantage of several knowledge bases, such as WordNet[3], Freebase[4] and YAGO[5], to automatically label free texts, like Wikipedia[6] and New York Times corpora[7], based on some heuristic alignment assumptions. An example accounting for the basic but practical assumption is illustrated in Figure 1, in which we know that the two entities (<Barack Obama, U.S.>) are not only involved in the *relation instances*[8] coming from knowledge bases (President-of(Barack Obama, U.S.) and Born-in(Barack Obama, U.S.)),

[1]http://www.itl.nist.gov/iaui/894.02/related projects/muc/
[2]http://www.itl.nist.gov/iad/mig/tests/ace/
[3]http://wordnet.princeton.edu
[4]http://www.freebase.com
[5]http://www.mpi-inf.mpg.de/yago-naga/yago
[6]http://www.wikipedia.org
[7]http://catalog.ldc.upenn.edu/LDC2008T19
[8]According to convention, we regard a structured triple $r(e_i, e_j)$ as a relation instance which is composed of a pair of entities $<e_i, e_j>$and a relation name r with respect to them.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 839–849,
Baltimore, Maryland, USA, June 23-25 2014.

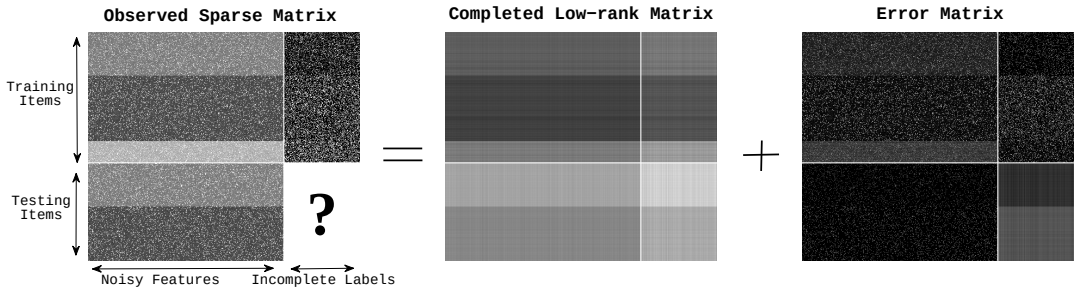


Figure 2: The procedure of noise-tolerant low-rank matrix completion. In this scenario, distantly supervised relation extraction task is transformed into completing the labels for testing items (entity pairs) in a sparse matrix that concatenates training and testing textual features with training labels. We seek to recover the underlying low-rank matrix and to complete the unknown testing labels simultaneously.

but also co-occur in several *relation mentions*[9] appearing in free texts (`Barack Obama is the 44th and current President of the U.S.` and `Barack Obama was born in Honolulu, Hawaii, U.S.`, etc.). We extract diverse textual features from all those *relation mentions* and combine them into a rich feature vector labeled by the *relation names* (`President-of` and `Born-in`) to produce a *weak* training corpus for relation classification.

This paradigm is promising to generate large-scale training corpora automatically. However, it comes up against three technical challeges:

- **Sparse features**. As we cannot tell what kinds of features are effective in advance, we have to use NLP toolkits, such as Stanford CoreNLP[10], to extract a variety of textual features, e.g., named entity tags, part-of-speech tags and lexicalized dependency paths. Unfortunately, most of them appear only once in the training corpus, and hence leading to very sparse features.

- **Noisy features**. Not all relation mentions express the corresponding relation instances. For example, the second relation mention in Figure 1 does not explicitly describe any relation instance, so features extracted from this sentence can be noisy. Such analogous cases commonly exist in feature extraction.

- **Incomplete labels**. Similar to noisy features, the generated labels can be incomplete. For example, the fourth relation mention in Figure 1 should have been labeled by the relation `Senate-of`. However, the incomplete knowledge base does not contain the corresponding relation instance (`Senate-of(Barack Obama, U.S.)`). Therefore, the distant supervision paradigm may generate incomplete labeling corpora.

In essence, distantly supervised relation extraction is an *incomplete* multi-label classification task with *sparse* and *noisy* features.

In this paper, we formulate the relation-extraction task from a novel perspective of using matrix completion with low rank criterion. To the best of our knowledge, we are the first to apply this technique on relation extraction with distant supervision. More specifically, as shown in Figure 2, we model the task with a sparse matrix whose rows present items (entity pairs) and columns contain noisy textual features and incomplete relation labels. In such a way, relation classification is transformed into a problem of completing the unknown labels for testing items in the sparse matrix that concatenates training and testing textual features with training labels, based on the assumption that the item-by-feature and item-by-label joint matrix is of low rank. The rationale of this assumption is that noisy features and incomplete labels are semantically correlated. The low-rank factorization of the sparse feature-label matrix delivers the low-dimensional representation of de-correlation for features and labels.

[9]The sentences that contain the given entity pair are called relation mentions.

[10]http://nlp.stanford.edu/downloads/corenlp.shtml

We contribute two optimization models, DRMC[11]-b and DRMC-1, aiming at exploiting the sparsity to recover the underlying low-rank matrix and to complete the unknown testing labels simultaneously. Moreover, the logistic cost function is integrated in our models to reduce the influence of noisy features and incomplete labels, due to that it is suitable for binary variables. We also modify the fixed point continuation (FPC) algorithm (Ma et al., 2011) to find the global optimum.

Experiments on two widely used datasets demonstrate that our noise-tolerant approaches outperform the baseline and the state-of-the-art methods. Furthermore, we discuss the influence of feature sparsity, and our approaches consistently achieve better performance than compared methods under different sparsity degrees.

2 Related Work

The idea of distant supervision was firstly proposed in the field of bioinformatics (Craven and Kumlien, 1999). Snow et al. (2004) used WordNet as the knowledge base to discover more hpyernym/hyponym relations between entities from news articles. However, either bioinformatic database or WordNet is maintained by a few experts, thus hardly kept up-to-date.

As we are stepping into the *big data* era, the explosion of unstructured Web texts simulates us to build more powerful models that can automatically extract relation instances from large-scale online natural language corpora without hand-labeled annotation. Mintz et al. (2009) adopted Freebase (Bollacker et al., 2008; Bollacker et al., 2007), a large-scale crowdsourcing knowledge base online which contains billions of relation instances and thousands of relation names, to *distantly supervise* Wikipedia corpus. The basic alignment assumption of this work is that if a pair of entities participate in a relation, *all sentences* that mention these entities are labeled by that relation name. Then we can extract a variety of textual features and learn a multi-class logistic regression classifier. Inspired by multi-instance learning (Maron and Lozano-Pérez, 1998), Riedel et al. (2010) relaxed the strong assumption and replaced *all sentences* with *at least one sentence*. Hoffmann et al. (2011) pointed out that many entity pairs have more than one relation. They extended the multi-instance learning framework (Riedel et al., 2010) to the multi-label circumstance. Surdeanu et al. (2012) proposed a novel approach to multi-instance multi-label learning for relation extraction, which jointly modeled all the sentences in texts and all labels in knowledge bases for a given entity pair. Other literatures (Takamatsu et al., 2012; Min et al., 2013; Zhang et al., 2013; Xu et al., 2013) addressed more specific issues, like how to construct the negative class in learning or how to adopt more information, such as name entity tags, to improve the performance.

Our work is more relevant to Riedel et al.'s (2013) which considered the task as a matrix factorization problem. Their approach is composed of several models, such as PCA (Collins et al., 2001) and collaborative filtering (Koren, 2008). However, they did not concern about the data noise brought by the basic assumption of distant supervision.

3 Model

We apply a new technique in the field of applied mathematics, i.e., low-rank matrix completion with convex optimization. The breakthrough work on this topic was made by Candès and Recht (2009) who proved that most low-rank matrices can be perfectly recovered from an incomplete set of entries. This promising theory has been successfully applied on many active research areas, such as computer vision (Cabral et al., 2011), recommender system (Rennie and Srebro, 2005) and system controlling (Fazel et al., 2001). Our models for relation extraction are based on the theoretic framework proposed by Goldberg et al. (2010), which formulated the multi-label transductive learning as a matrix completion problem. The new framework for classification enhances the robustness to data noise by penalizing different cost functions for features and labels.

3.1 Formulation

Suppose that we have built a training corpus for relation classification with n items (entity pairs), d-dimensional textual features, and t labels (relations), based on the basic alignment assumption proposed by Mintz et al. (2009). Let $X_{train} \in \mathbb{R}^{n\times d}$ and $Y_{train} \in \mathbb{R}^{n\times t}$ denote the feature matrix and the label matrix for training, respectively. The linear classifier we adopt aims to explicitly learn the weight matrix $\mathbf{W} \in \mathbb{R}^{d\times t}$ and the bias column

[11] It is the abbreviation for **D**istant supervision for **R**elation extraction with **M**atrix **C**ompletion

vector $\mathbf{b} \in \mathbb{R}^{t\times 1}$ with the constraint of minimizing the loss function l,

$$\arg\min_{\mathbf{W},\mathbf{b}} \; l(Y_{train}, \left[\begin{array}{cc} \mathbf{1} & X_{train} \end{array}\right] \left[\begin{array}{c} \mathbf{b}^T \\ \mathbf{W} \end{array}\right]), \quad (1)$$

where $\mathbf{1}$ is the all-one column vector. Then we can predict the label matrix $Y_{test} \in \mathbb{R}^{m\times t}$ of m testing items with respect to the feature matrix $X_{test} \in \mathbb{R}^{m\times d}$. Let

$$\mathbf{Z} = \left[\begin{array}{cc} X_{train} & Y_{train} \\ X_{test} & Y_{test} \end{array}\right].$$

This linear classification problem can be transformed into completing the unobservable entries in Y_{test} by means of the observable entries in X_{train}, Y_{train} and X_{test}, based on the assumption that the rank of matrix $\mathbf{Z} \in \mathbb{R}^{(n+m)\times(d+t)}$ is low. The model can be written as,

$$\begin{aligned} &\arg\min_{\mathbf{Z}\in\mathbb{R}^{(n+m)\times(d+t)}} \quad \text{rank}(\mathbf{Z}) \\ &\text{s.t. } \forall(i,j)\in\Omega_X, \;\; z_{ij} = x_{ij}, \\ &\qquad (1\le i\le n+m, \;\; 1\le j\le d), \\ &\qquad \forall(i,j)\in\Omega_Y, \;\; z_{i(j+d)} = y_{ij}, \\ &\qquad (1\le i\le n, \;\; 1\le j\le t), \end{aligned} \quad (2)$$

where we use Ω_X to represent the index set of observable feature entries in X_{train} and X_{test}, and Ω_Y to denote the index set of observable label entries in Y_{train}.

Formula (2) is usually impractical for real problems as the entries in the matrix $\mathbf{Z}$ are corrupted by noise. We thus define

$$\mathbf{Z} = \mathbf{Z}^* + \mathbf{E},$$

where $\mathbf{Z}^*$ as the underlying low-rank matrix

$$\mathbf{Z}^* = \left[\begin{array}{cc} X^* & Y^* \end{array}\right] = \left[\begin{array}{cc} X^*_{train} & Y^*_{train} \\ X^*_{test} & Y^*_{test} \end{array}\right],$$

and $\mathbf{E}$ is the error matrix

$$\mathbf{E} = \left[\begin{array}{cc} E_{X_{train}} & E_{Y_{train}} \\ E_{X_{test}} & 0 \end{array}\right].$$

The rank function in Formula (2) is a non-convex function that is difficult to be optimized. The surrogate of the function can be the convex nuclear norm $||\mathbf{Z}||_* = \sum\sigma_k(\mathbf{Z})$ (Candès and Recht, 2009), where σ_k is the k-th largest singular value of $\mathbf{Z}$. To tolerate the noise entries in the error matrix $\mathbf{E}$, we minimize the cost functions C_x and C_y for features and labels respectively, rather than using the hard constraints in Formula (2).

According to Formula (1), $\mathbf{Z}^* \in \mathbb{R}^{(n+m)\times(d+t)}$ can be represented as $[X^*, \mathbf{W}X^*]$ instead of $[X^*, Y^*]$, by explicitly modeling the bias vector $\mathbf{b}$. Therefore, this convex optimization model is called **DRMC-b**,

$$\begin{aligned} \arg\min_{\mathbf{Z},\mathbf{b}} \;\; &\mu||\mathbf{Z}||_* + \frac{1}{|\Omega_X|}\sum_{(i,j)\in\Omega_X} C_x(z_{ij}, x_{ij}) \\ &+ \frac{\lambda}{|\Omega_Y|}\sum_{(i,j)\in\Omega_Y} C_y(z_{i(j+d)} + b_j, y_{ij}), \end{aligned} \quad (3)$$

where μ and λ are the positive trade-off weights. More specifically, we minimize the nuclear norm $||\mathbf{Z}||_*$ via employing the regularization terms, i.e., the cost functions C_x and C_y for features and labels.

If we implicitly model the bias vector $\mathbf{b}$, $\mathbf{Z}^* \in \mathbb{R}^{(n+m)\times(1+d+t)}$ can be denoted by $[\mathbf{1}, X^*, \mathbf{W}'X^*]$ instead of $[X^*, Y^*]$, in which $\mathbf{W}'$ takes the role of $[\mathbf{b}^T; \mathbf{W}]$ in DRMC-b. Then we derive another optimization model called **DRMC-1**,

$$\begin{aligned} \arg\min_{\mathbf{Z}} \;\; &\mu||\mathbf{Z}||_* + \frac{1}{|\Omega_X|}\sum_{(i,j)\in\Omega_X} C_x(z_{i(j+1)}, x_{ij}) \\ &+ \frac{\lambda}{|\Omega_Y|}\sum_{(i,j)\in\Omega_Y} C_y(z_{i(j+d+1)}, y_{ij}) \\ \text{s.t.} \;\; &\mathbf{Z}(:,1) = \mathbf{1}, \end{aligned} \quad (4)$$

where $\mathbf{Z}(:,1)$ denotes the first column of $\mathbf{Z}$.

For our relation classification task, both features and labels are binary. We assume that the actual entry u belonging to the underlying matrix $\mathbf{Z}^*$ is randomly generated via a sigmoid function (Jordan, 1995): $Pr(u|v) = 1/(1+e^{-uv})$, given the observed binary entry v from the observed sparse matrix $\mathbf{Z}$. Then, we can apply the log-likelihood cost function to measure the conditional probability and derive the *logistic cost function* for C_x and C_y,

$$C(u,v) = -\log Pr(u|v) = \log(1+e^{-uv}),$$

After completing the entries in Y_{test}, we adopt the sigmoid function to calculate the conditional probability of relation r_j, given entity pair p_i pertaining to y_{ij} in Y_{test},

$$Pr(r_j|p_i) = \frac{1}{1+e^{-y_{ij}}}, \quad y_{ij}\in Y_{test}.$$

Finally, we can achieve Top-N predicted relation instances via ranking the values of $Pr(r_j|p_i)$.

4 Algorithm

The matrix rank minimization problem is NP-hard. Therefore, Candés and Recht (2009) suggested to use a convex relaxation, the nuclear norm minimization instead. Then, Ma et al. (2011) proposed the fixed point continuation (FPC) algorithm which is fast and robust. Moreover, Goldfrab and Ma (2011) proved the convergence of the FPC algorithm for solving the nuclear norm minimization problem. We thus adopt and modify the algorithm aiming to find the optima for our noise-tolerant models, i.e., Formulae (3) and (4).

4.1 Fixed point continuation for DRMC-b

Algorithm 1 describes the modified FPC algorithm for solving DRMC-b, which contains two steps for each iteration,

Gradient step: In this step, we infer the matrix gradient $g(\mathbf{Z})$ and bias vector gradient $g(\mathbf{b})$ as follows,

$$g(z_{ij}) = \begin{cases} \frac{1}{|\Omega_X|}\frac{-x_{ij}}{1+e^{x_{ij}z_{ij}}}, & (i,j) \in \Omega_X \\ \frac{\lambda}{|\Omega_Y|}\frac{-y_{i(j-d)}}{1+e^{y_{i(j-d)}(z_{ij}+b_j)}}, & (i,j-d) \in \Omega_Y \\ 0, & otherwise \end{cases}$$

and

$$g(b_j) = \frac{\lambda}{|\Omega_Y|}\sum_{i:(i,j)\in\Omega_Y}\frac{-y_{ij}}{1+e^{y_{ij}(z_{i(j+d)}+b_j)}}.$$

We use the gradient descents $\mathbf{A} = \mathbf{Z} - \tau_z g(\mathbf{Z})$ and $\mathbf{b} = \mathbf{b} - \tau_b g(\mathbf{b})$ to gradually find the global minima of the cost function terms in Formula (3), where τ_z and τ_b are step sizes.

Shrinkage step: The goal of this step is to minimize the nuclear norm $||\mathbf{Z}||_*$ in Formula (3). We perform the singular value decomposition (SVD) (Golub and Kahan, 1965) for $\mathbf{A}$ at first, and then cut down each singular value. During the iteration, any negative value in $\mathbf{\Sigma} - \tau_{\mathbf{z}}\mu$ is assigned by zero, so that the rank of reconstructed matrix $\mathbf{Z}$ will be reduced, where $\mathbf{Z} = \mathbf{U}max(\mathbf{\Sigma} - \tau_{\mathbf{z}}\mu, 0)\mathbf{V^T}$.

To accelerate the convergence, we use a continuation method to improve the speed. μ is initialized by a large value μ_1, thus resulting in the fast reduction of the rank at first. Then the convergence slows down as μ decreases while obeying $\mu_{k+1} = max(\mu_k\eta_\mu, \mu_F)$. μ_F is the final value of μ, and η_μ is the decay parameter.

For the stopping criteria in inner iterations, we define the *relative error* to measure the residual of matrix $\mathbf{Z}$ between two successive iterations,

Algorithm 1 FPC algorithm for solving DRMC-b

Input:
Initial matrix $\mathbf{Z_0}$, bias $\mathbf{b_0}$; Parameters μ, λ;
Step sizes τ_z, τ_b.

Set $\mathbf{Z} = \mathbf{Z_0}$, $\mathbf{b} = \mathbf{b_0}$.
foreach $\mu = \mu_1 > \mu_2 > ... > \mu_F$ **do**
  while relative error $> \varepsilon$ **do**
    Gradient step:
    $\mathbf{A} = \mathbf{Z} - \tau_z g(\mathbf{Z}), \mathbf{b} = \mathbf{b} - \tau_b g(\mathbf{b})$.
    Shrinkage step:
    $\mathbf{U\Sigma V^T} = \text{SVD}(\mathbf{A})$,
    $\mathbf{Z} = \mathbf{U}\ max(\mathbf{\Sigma} - \tau_{\mathbf{z}}\mu, 0)\ \mathbf{V^T}$.
  end while
end foreach

Output: Completed Matrix $\mathbf{Z}$, bias $\mathbf{b}$.

$$\frac{||\mathbf{Z}^{k+1} - \mathbf{Z}^k||_F}{max(1, ||\mathbf{Z}^k||_F)} \leq \varepsilon,$$

where ε is the convergence threshold.

4.2 Fixed point continuation for DRMC-1

Algorithm 2 is similar to Algorithm 1 except for two differences. First, there is no bias vector $\mathbf{b}$. Second, a projection step is added to enforce the first column of matrix $\mathbf{Z}$ to be $\mathbf{1}$. In addition, The matrix gradient $g(\mathbf{Z})$ for DRMC-1 is

$$g(z_{ij}) = \begin{cases} \frac{1}{|\Omega_X|}\frac{-x_{i(j-1)}}{1+e^{x_{i(j-1)}z_{ij}}}, & (i,j-1) \in \Omega_X \\ \frac{\lambda}{|\Omega_Y|}\frac{-y_{i(j-d-1)}}{1+e^{y_{i(j-d-1)}z_{ij}}}, & (i,j-d-1) \in \Omega_Y \\ 0, & otherwise \end{cases}.$$

Algorithm 2 FPC algorithm for solving DRMC-1

Input:
Initial matrix $\mathbf{Z_0}$; Parameters μ, λ;
Step sizes τ_z.

Set $\mathbf{Z} = \mathbf{Z_0}$.
foreach $\mu = \mu_1 > \mu_2 > ... > \mu_F$ **do**
  while relative error $> \varepsilon$ **do**
    Gradient step: $\mathbf{A} = \mathbf{Z} - \tau_z g(\mathbf{Z})$.
    Shrinkage step:
    $\mathbf{U\Sigma V^T} = \text{SVD}(\mathbf{A})$,
    $\mathbf{Z} = \mathbf{U}\ max(\mathbf{\Sigma} - \tau_{\mathbf{z}}\mu, 0)\ \mathbf{V^T}$.
    Projection step: $\mathbf{Z}(:,1) = \mathbf{1}$.
  end while
end foreach

Output: Completed Matrix $\mathbf{Z}$.

Dataset	# of training tuples	# of testing tuples	% with more than one label	# of features	# of relation labels
NYT'10	4,700	1,950	7.5%	244,903	51
NYT'13	8,077	3,716	0%	1,957	51

Table 1: Statistics about the two widely used datasets.

Model	NYT'10 (θ=2)	NYT'10 (θ=3)	NYT'10 (θ=4)	NYT'10 (θ=5)	NYT'13
DRMC-b	51.4 $\pm$ 8.7 (51)	45.6 $\pm$ 3.4 (46)	41.6 $\pm$ 2.5 (43)	36.2 $\pm$ 8.8(37)	84.6 $\pm$ 19.0 (85)
DRMC-1	16.0 $\pm$ 1.0 (16)	16.4 $\pm$ 1.1(17)	16 $\pm$ 1.4 (17)	16.8 $\pm$ 1.5(17)	15.8 $\pm$ 1.6 (16)

Table 2: The range of optimal ranks for DRMC-b and DRMC-1 through five-fold cross validation. The threshold θ means filtering the features that appear less than θ times. The values in brackets pertaining to DRMC-b and DRMC-1 are the exact optimal ranks that we choose for the completed matrices on testing sets.

5 Experiments

In order to conduct reliable experiments, we adjust and estimate the parameters for our approaches, DRMC-b and DRMC-1, and compare them with other four kinds of landmark methods (Mintz et al., 2009; Hoffmann et al., 2011; Surdeanu et al., 2012; Riedel et al., 2013) on two public datasets.

5.1 Dataset

The two widely used datasets that we adopt are both automatically generated by aligning Freebase to New York Times corpora. The first dataset[12], NYT'10, was developed by Riedel et al. (2010), and also used by Hoffmann et al. (2011) and Surdeanu et al. (2012). Three kinds of features, namely, lexical, syntactic and named entity tag features, were extracted from relation mentions. The second dataset[13], NYT'13, was also released by Riedel et al. (2013), in which they only regarded the lexicalized dependency path between two entities as features. Table 1 shows that the two datasets differ in some main attributes. More specifically, NYT'10 contains much higher dimensional features than NYT'13, whereas fewer training and testing items.

5.2 Parameter setting

In this part, we address the issue of setting parameters: the trade-off weights μ and λ, the step sizes τ_z and τ_b, and the decay parameter η_μ.

We set $\lambda = 1$ to make the contribution of the cost function terms for feature and label matrices equal in Formulae (3) and (4). μ is assigned by a series of values obeying $\mu_{k+1} = max(\mu_k \eta_\mu, \mu_F)$. We follow the suggestion in (Goldberg et al., 2010) that μ starts at $\sigma_1 \eta_\mu$, and σ_1 is the largest singular value of the matrix $\mathbf{Z}$. We set $\eta_\mu = 0.01$. The final value of μ, namely μ_F, is equal to 0.01. Ma et al. (2011) revealed that as long as the non-negative step sizes satisfy $\tau_z < min(\frac{4|\Omega_Y|}{\lambda}, |\Omega_X|)$ and $\tau_b < \frac{4|\Omega_Y|}{\lambda(n+m)}$, the FPC algorithm will guarantee to converge to a global optimum. Therefore, we set $\tau_z = \tau_b = 0.5$ to satisfy the above constraints on both two datasets.

5.3 Rank estimation

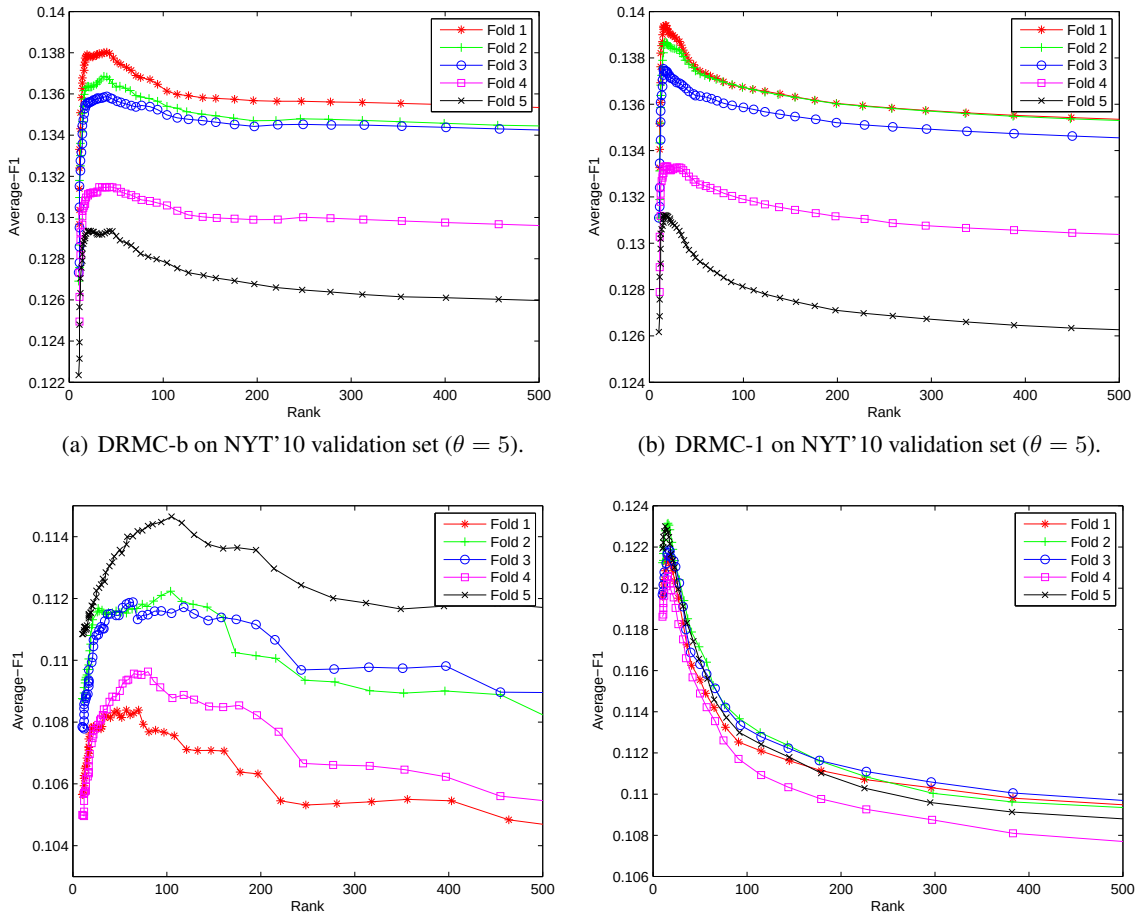
Even though the FPC algorithm converges in iterative fashion, the value of ε varying with different datasets is difficult to be decided. In practice, we record the rank of matrix $\mathbf{Z}$ at each round of iteration until it converges at a rather small threshold $\varepsilon = 10^{-4}$. The reason is that we suppose the optimal low-rank representation of the matrix $\mathbf{Z}$ conveys the truly effective information about underlying semantic correlation between the features and the corresponding labels.

We use the five-fold cross validation on the validation set and evaluate the performance on each fold with different ranks. At each round of iteration, we gain a recovered matrix and average the F1[14] scores from Top-5 to Top-all predicted relation instances to measure the performance. Figure 3 illustrates the curves of average F1 scores. After recording the rank associated with the highest F1 score on each fold, we compute the mean and the standard deviation to estimate the range of optimal rank for testing. Table 2 lists the range of optimal ranks for DRMC-b and DRMC-1 on NYT'10 and NYT'13.

[12] http://iesl.cs.umass.edu/riedel/ecml/

[13] http://iesl.cs.umass.edu/riedel/data-univSchema/

[14] $F1 = \frac{2 \times precision \times recall}{precision + recall}$



(a) DRMC-b on NYT'10 validation set ($\theta = 5$).

(b) DRMC-1 on NYT'10 validation set ($\theta = 5$).

(c) DRMC-b on NYT'13 validation set.

(d) DRMC-1 on NYT'13 validation set.

Figure 3: Five-fold cross validation for rank estimation on two datasets.

On both two datasets, we observe an identical phenomenon that the performance gradually increases as the rank of the matrix declines before reaching the optimum. However, it sharply decreases if we continue reducing the optimal rank. An intuitive explanation is that the high-rank matrix contains much noise and the model tends to be overfitting, whereas the matrix of excessively low rank is more likely to lose principal information and the model tends to be underfitting.

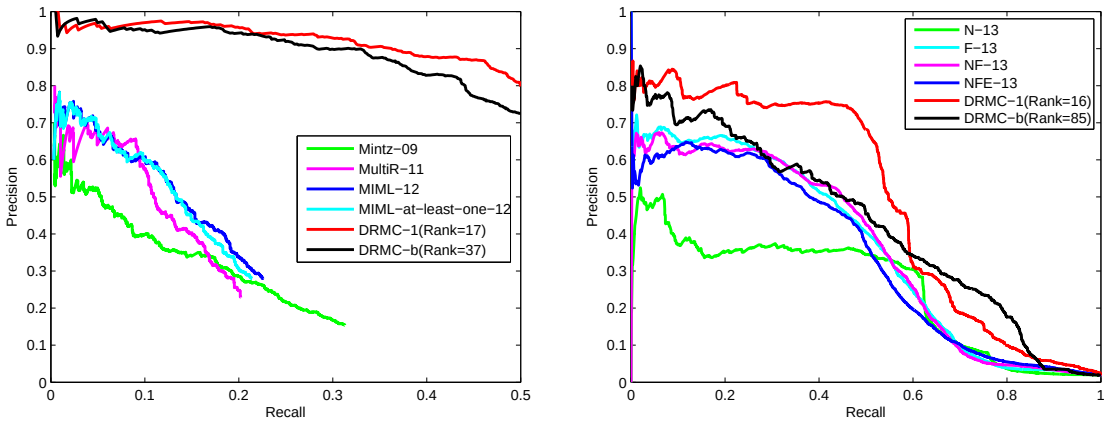
5.4 Method Comparison

Firstly, we conduct experiments to compare our approaches with Mintz-09 (Mintz et al., 2009), MultiR-11 (Hoffmann et al., 2011), MIML-12 and MIML-at-least-one-12 (Surdeanu et al., 2012) on NYT'10 dataset. Surdeanu et al. (2012) released the open source code[15] to reproduce the experimental results on those previous methods. Moreover, their programs can control the feature sparsity degree through a threshold θ which filters the features that appears less than θ times. They set $\theta = 5$ in the original code by default. Therefore, we follow their settings and adopt the same way to filter the features. In this way, we guarantee the fair comparison for all methods. Figure 4 (a) shows that our approaches achieve the significant improvement on performance.

We also perform the experiments to compare our approaches with the state-of-the-art NFE-13[16] (Riedel et al., 2013) and its sub-methods (N-13, F-13 and NF-13) on NYT'13 dataset. Figure 4 (b) illustrates that our approaches still outperform the state-of-the-art methods. In practical applications, we also concern about the precision on Top-N predicted relation instances. Therefore, We compare the precision of Top-100s, Top-200s and Top-500s for DRMC-1, DRMC-b and the state-of-the-

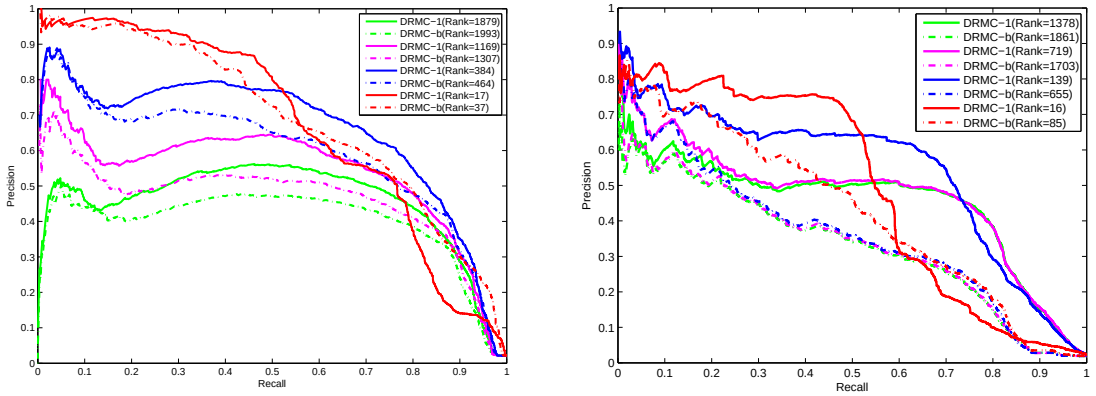
[15] http://nlp.stanford.edu/software/mimlre.shtml

[16] Readers may refer to the website, http://www.riedelcastro.org/uschema for the details of those methods. We bypass the description due to the limitation of space.



(a) NYT'10 testing set ($\theta = 5$). (b) NYT'13 testing set.

Figure 4: Method comparison on two testing sets.



(a) NYT'10 testing set ($\theta = 5$). (b) NYT'13 testing set.

Figure 5: Precision-Recall curve for DRMC-b and DRMC-1 with different ranks on two testing sets.

Top-N	NFE-13	DRMC-b	DRMC-1
Top-100	62.9%	**82.0%**	80.0%
Top-200	57.1%	77.0%	**80.0%**
Top-500	37.2%	70.2%	**77.0%**
Average	52.4%	76.4%	**79.0%**

Table 3: Precision of NFE-13, DRMC-b and DRMC-1 on Top-100, Top-200 and Top-500 predicted relation instances.

art method NFE-13 (Riedel et al., 2013). Table 3 shows that DRMC-b and DRMC-1 achieve 24.0% and 26.6% precision increments on average, respectively.

6 Discussion

We have mentioned that the basic alignment assumption of distant supervision (Mintz et al., 2009) tends to generate noisy (noisy features and incomplete labels) and sparse (sparse features) data. In this section, we discuss how our approaches tackle these natural flaws.

Due to the noisy features and incomplete labels, the underlying low-rank data matrix with truly effective information tends to be corrupted and the rank of observed data matrix can be extremely high. Figure 5 demonstrates that the ranks of data matrices are approximately 2,000 for the initial optimization of DRMC-b and DRMC-1. However, those high ranks result in poor performance. As the ranks decline before approaching the optimum, the performance gradually improves, implying that our approaches filter the noise in data and keep the principal information for classification via recovering the underlying low-rank data matrix.

Furthermore, we discuss the influence of the feature sparsity for our approaches and the state-

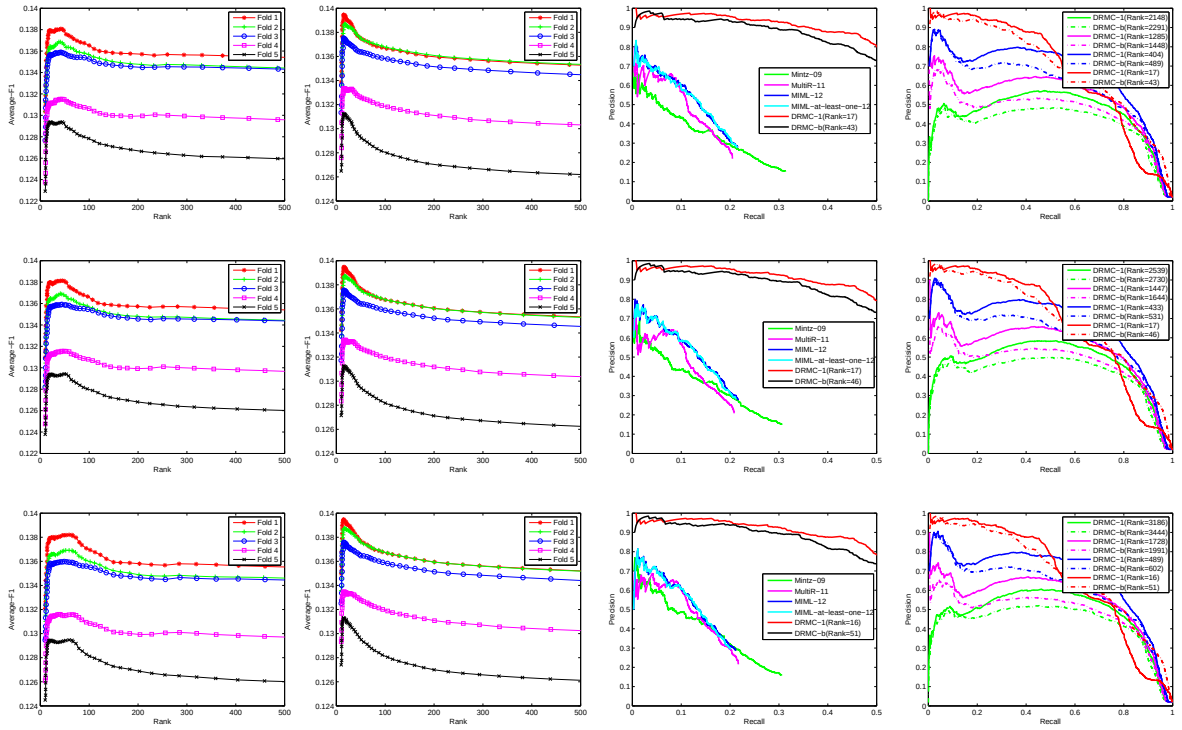


Figure 6: Feature sparsity discussion on NYT'10 testing set. Each row (from top to bottom, $\theta = 4, 3, 2$) illustrates a suite of experimental results. They are, from left to right, five-fold cross validation for rank estimation on DRMC-b and DRMC-1, method comparison and precision-recall curve with different ranks, respectively.

of-the-art methods. We relax the feature filtering threshold ($\theta = 4, 3, 2$) in Surdeanu et al.'s (2012) open source program to generate more sparse features from NYT'10 dataset. Figure 6 shows that our approaches consistently outperform the baseline and the state-of-the-art methods with diverse feature sparsity degrees. Table 2 also lists the range of optimal rank for DRMC-b and DRMC-1 with different θ. We observe that for each approach, the optimal range is relatively stable. In other words, for each approach, the amount of truly effective information about underlying semantic correlation keeps constant for the same dataset, which, to some extent, explains the reason why our approaches are robust to sparse features.

7 Conclusion and Future Work

In this paper, we contributed two noise-tolerant optimization models[17], DRMC-b and DRMC-1, for distantly supervised relation extraction task from a novel perspective. Our models are based on matrix completion with low-rank criterion. Experiments demonstrated that the low-rank representation of the feature-label matrix can exploit the underlying semantic correlated information for relation classification and is effective to overcome the difficulties incurred by sparse and noisy features and incomplete labels, so that we achieved significant improvements on performance.

Our proposed models also leave open questions for distantly supervised relation extraction task. First, they can not process new coming testing items efficiently, as we have to reconstruct the data matrix containing not only the testing items but also all the training items for relation classification, and compute in iterative fashion again. Second, the volume of the datasets we adopt are relatively small. For the future work, we plan to improve our models so that they will be capable of incremental learning on large-scale datasets (Chang, 2011).

[17]The source code can be downloaded from `https://github.com/nlpgeek/DRMC/tree/master`

Acknowledgments

This work is supported by National Program on Key Basic Research Project (973 Program) under Grant 2013CB329304, National Science Foundation of China (NSFC) under Grant No.61373075.

References

Nguyen Bach and Sameer Badaskar. 2007. A review of relation extraction. *Literature review for Language and Statistics II*.

Kurt Bollacker, Robert Cook, and Patrick Tufts. 2007. Freebase: A shared database of structured general human knowledge. In *Proceedings of the national conference on Artificial Intelligence*, volume 22, page 1962. AAAI Press.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM.

Ricardo S Cabral, Fernando Torre, João P Costeira, and Alexandre Bernardino. 2011. Matrix completion for multi-label image classification. In *Advances in Neural Information Processing Systems*, pages 190–198.

Emmanuel J Candès and Benjamin Recht. 2009. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772.

Edward Y Chang. 2011. *Foundations of Large-Scale Multimedia Information Management and Retrieval*. Springer.

Michael Collins, Sanjoy Dasgupta, and Robert E Schapire. 2001. A generalization of principal components analysis to the exponential family. In *Advances in neural information processing systems*, pages 617–624.

Mark Craven and Johan Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In *ISMB*, volume 1999, pages 77–86.

Maryam Fazel, Haitham Hindi, and Stephen P Boyd. 2001. A rank minimization heuristic with application to minimum order system approximation. In *American Control Conference, 2001. Proceedings of the 2001*, volume 6, pages 4734–4739. IEEE.

Andrew Goldberg, Ben Recht, Junming Xu, Robert Nowak, and Xiaojin Zhu. 2010. Transduction with matrix completion: Three birds with one stone. In *Advances in neural information processing systems*, pages 757–765.

Donald Goldfarb and Shiqian Ma. 2011. Convergence of fixed-point continuation algorithms for matrix rank minimization. *Foundations of Computational Mathematics*, 11(2):183–210.

Gene Golub and William Kahan. 1965. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial & Applied Mathematics, Series B: Numerical Analysis*, 2(2):205–224.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 541–550, Portland, Oregon, USA, June. Association for Computational Linguistics.

Michael Jordan. 1995. Why the logistic function? a tutorial discussion on probabilities and neural networks. *Computational Cognitive Science Technical Report*.

Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM.

Shiqian Ma, Donald Goldfarb, and Lifeng Chen. 2011. Fixed point and bregman iterative methods for matrix rank minimization. *Mathematical Programming*, 128(1-2):321–353.

Oded Maron and Tomás Lozano-Pérez. 1998. A framework for multiple-instance learning. *Advances in neural information processing systems*, pages 570–576.

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 777–782, Atlanta, Georgia, June. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Jasson DM Rennie and Nathan Srebro. 2005. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd international conference on Machine learning*, pages 713–719. ACM.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*,

pages 74–84, Atlanta, Georgia, June. Association for Computational Linguistics.

Rion Snow, Daniel Jurafsky, and Andrew Y Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. *Advances in Neural Information Processing Systems 17*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465. Association for Computational Linguistics.

Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. 2012. Reducing wrong labels in distant supervision for relation extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 721–729. Association for Computational Linguistics.

Wei Xu, Raphael Hoffmann, Le Zhao, and Ralph Grishman. 2013. Filling knowledge base gaps for distant supervision of relation extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 665–670, Sofia, Bulgaria, August. Association for Computational Linguistics.

Xingxing Zhang, Jianwen Zhang, Junyu Zeng, Jun Yan, Zheng Chen, and Zhifang Sui. 2013. Towards accurate distant supervision for relational facts extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 810–815, Sofia, Bulgaria, August. Association for Computational Linguistics.

Guodong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 427–434. Association for Computational Linguistics.

Enhancing Grammatical Cohesion: Generating Transitional Expressions for SMT

Mei Tu **Yu Zhou** **Chengqing Zong**
National Laboratory of Pattern Recognition,
Institute of Automation,
Chinese Academy of Sciences
{mtu,yzhou,cqzong}@nlpr.ia.ac.cn

Abstract

Transitional expressions provide glue that holds ideas together in a text and enhance the logical organization, which together help improve readability of a text. However, in most current statistical machine translation (SMT) systems, the outputs of compound-complex sentences still lack proper transitional expressions. As a result, the translations are often hard to read and understand. To address this issue, we propose two novel models to encourage generating such transitional expressions by introducing the source compound-complex sentence structure (CSS). Our models include a CSS-based translation model, which generates new CSS-based translation rules, and a generative transfer model, which encourages producing transitional expressions during decoding. The two models are integrated into a hierarchical phrase-based translation system to evaluate their effectiveness. The experimental results show that significant improvements are achieved on various test data meanwhile the translations are more cohesive and smooth.

1 Introduction

During the last decade, great progress has been made on statistical machine translation (SMT) models. However, these translations still suffer from poor readability, especially translations of compound-complex sentences. One of the main reasons may be that most existing models concentrate more on producing well-translated local sentence fragments, but largely ignore global cohesion between the fragments. Generally, cohesion, including lexical and grammatical cohesion, contributes much to the understandability and smoothness of a text.

Recently, researchers have begun addressing the lexical cohesion of SMT (Gong et al., 2011; Xiao et al., 2011; Wong and Kit, 2012; Xiong, 2013). These efforts focus mainly on the co-occurrence of lexical items in a similar environment. Grammatical cohesion[1] (Halliday and Hassan, 1976) in SMT has been little mentioned in previous work. Translations without grammatical cohesion is hard to read, mostly due to loss of cohesive and transitional expressions between two sentence fragments. Thus, generating transitional expressions is necessary for achieving grammatical cohesion. However, it is not easy to produce such transitional expressions in SMT. As an example, consider the Chinese-to-English translation in Figure 1.

Source Chinese sentence:

[尽管 减轻 污染 的 呼声 不断 ，]1 [公众
Although reduce pollution of calls continue , public

日渐 愤怒 ，]2 [污染 还是 变得 更 糟糕
growing angry , pollution still become more worse

了 ，]3 [越发 显出 环保 的 紧迫性 。]4
already , more show environment protection of urgent .

Target English golden translation:

***Despite** frequent calls for cutting pollution, **and** growing public anger, the problem has only got worse, **which** increasingly shows the urgency of environmental protection.*

Figure 1: An example of Chinese-to-English translation. The English translation sentence has three transitional phrases: ***Despite, and, which***.

There are 4 sub-sentences separated by commas in the Chinese sentence. We have tried to translate the Chinese sentence using many well-

[1] Grammatical cohesion can make relations among sentences more explicit. There are various grammatically cohesive devices (reference, substitution ellipsis and conjunction) that tie fragments together in a cohesive way.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 850–860,
Baltimore, Maryland, USA, June 23-25 2014.

known online translators, but find that it is very difficult to generate the target transitional expressions, especially when there is no explicit connective word in the source sentence, such as generating "***and*** " and "***which***" in Figure 1.

Fortunately, the functional relationships between two neighboring source sub-sentences provide us with a good perspective and the inspiration to generate those transitional phrases. Figure 1 shows that the first and the second Chinese sub-sentences form a ***parallel*** relation. Thus, even though there is no distinct connective word at the beginning of the second source sub-sentence, a good translator is still able to insert or generate an "***and***" as a connection word to make the target translation more cohesive.

Based on the above analysis, this paper focuses on the target grammatical cohesion in SMT to make the translation more understandable, especially for languages with great difference in linguistic structure like Chinese and English. To the best of our knowledge, our work is the first attempt to generate target transitional expressions for SMT grammatical cohesion by introducing the functional relationships of source sentences. In this work, we propose two models. One is a new translation model that is utilized to generate new translation rules combined with the information of source functional relationships. The other is a generative transfer model that encourages producing transitional phrases during decoding. Our experimental results on Chinese-to-English translation demonstrate that the translation readability is greatly improved by introducing the cohesive information.

The remainder of the paper is organized as follows. In Section 2, we describe the functional relationships of Chinese compound-complex sentences. In Section 3, we present our models and show how to integrate the models into an SMT system. Our experimental results are reported in Section 4. A survey of related work is conducted in Section 5, and we conclude our work and outline the future work in Section 6.

2 Chinese Compound-Complex Sentence Structure

To acquire the functional relationships of a Chinese compound-complex sentence, Zhou (2004) proposed a well-annotated scheme to build the Compound-complex Sentence Structure (**CSS**). The structure explicitly shows the minimal semantic spans, called elementary units (***eu***s), and also depicts the hierarchical relations among ***eu***s. There are 11 common types of functional relationships [2] annotated in the Tsinghua Chinese Treebank (Zhou, 2004).

Under the annotation scheme of the Tsinghua Chinese Treebank, the Chinese sentence of example in Figure 1 is represented as the tree shown in Figure 2. In this example, each sub-sentence is an ***eu***. $\boldsymbol{eu_1}$ and $\boldsymbol{eu_2}$ are combined with a *parallel* relationship, followed by $\boldsymbol{eu_3}$ with an *adversative* relationship. $\boldsymbol{eu_1}$, $\boldsymbol{eu_2}$, and $\boldsymbol{eu_3}$ form a large semantic span[3], connected with $\boldsymbol{eu_4}$ by a *consequence* relationship. All of the ***eu***s are organized into various functional relationships and finally form a hierarchical tree.

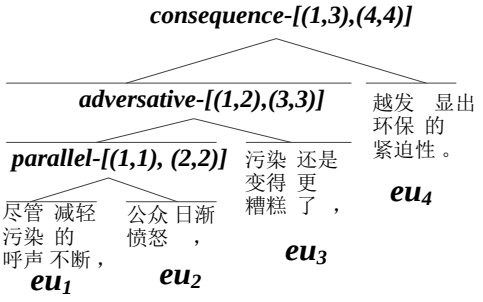


Figure 2: The compound-complex sentence structure of the Chinese sentence in Figure 1.

Formally, given a compound-complex sentence structure (CSS), each node in the CSS can be represented as a tuple $R-[(s_1,e_1),...(s_l,e_l),...,(s_L,e_L)]$. R represents the relationship, which has L children. For each child of R, a pair (s_l,e_l) records its *start* and *end* ***eu***s. For example, ***adversative-[(1,2), (3,3)]*** in Figure 2 means that two children are controlled by the relationship ***adversative***, and the left child consists of $\boldsymbol{eu_1}$ and $\boldsymbol{eu_2}$, while the right child contains only $\boldsymbol{eu_3}$.

CSS has much in common with Rhetorical Structure (Mann and Thompson, 1988) in English, which also describe the semantic relation between discourse units. But the Rhetorical Structure involves much richer relations on the document-level, and little corpus is open for Chinese.

In the following, we will describe in detail how to utilize such CSS information for modelling in SMT.

[2] They are *parallel, consequence, progressive, alternative, causal, purpose, hypothesis, condition, adversative, explanation,* and *flowing* relationships.

[3] A semantic span can include one or more ***eu***s.

3 Modelling

Our purpose is to enhance the grammatical cohesion by exploiting the source CSS information. Therefore, theoretically, the conditional probability of a target translation $\boldsymbol{e}_s$ conditioned on the source CSS-based tree $\boldsymbol{f}_t$ is given by $P(\boldsymbol{e}_s \mid \boldsymbol{f}_t)$, and the final translation $\hat{\boldsymbol{e}}_s$ is obtained with the following formula:

$$\hat{\boldsymbol{e}}_s = \arg\max_{\boldsymbol{e}_s}\{\mathrm{P}(\boldsymbol{e}_s \mid \boldsymbol{f}_t)\} \quad (1)$$

Following Och and Ney (2002), our model is framed as a log-linear model:

$$P(\boldsymbol{e}_s \mid \boldsymbol{f}_t) = \frac{\exp\sum_k \lambda_k h_k(\boldsymbol{e}_s, \boldsymbol{f}_t)}{\sum_{\boldsymbol{e}'_s} \exp\sum_k \lambda_k h_k(\boldsymbol{e}'_s, \boldsymbol{f}_t)} \quad (2)$$

where $h(\boldsymbol{e}_s, \boldsymbol{f}_t)$ is a feature with weight λ. Then, the best translation is:

$$\hat{\boldsymbol{e}}_s = \arg\max_{\boldsymbol{e}_s} \exp\sum_k \lambda_k h_k(\boldsymbol{e}_s, \boldsymbol{f}_t) \quad (3)$$

Our models make use of CSS with two strategies:

1) **CSS-based translation model**: following formula (1), we obtain the cohesion information by modifying the translation rules with their probabilities $P(\boldsymbol{e}_s \mid \boldsymbol{f}_t)$ based on word alignments between the source CSS-tree and the target string;

2) **CSS-based transfer model:** following formula (3), we introduce a transfer score to encourage the decoder to generate transitional words and phrases; the score is utilized as an additional feature $h_k(\boldsymbol{e}_s, \boldsymbol{f}_t)$ in the log-linear model.

3.1 CSS-based Translation Model

For the existing translation models, the entire training process is conducted at the lexical or syntactic level without grammatically cohesive information. As a result, it is difficult to utilize such cohesive information during decoding. Instead, we reserve the cohesive information in the training process by converting the original source sentence into tagged-flattened CSS and then perform word alignment and extract the translation rules from the bilingual flattened source CSS and the target string.

As introduced in Section 2, a CSS consists of nodes, and a node can be represented as a tuple $R-[(s_1,e_1),...(s_l,e_l),...,(s_L,e_L)]$. In this representation, the relationship R is the most important factor because different relationships directly reflect different cohesive expressions. In addition, the children's positions always play a strong role in choosing cohesive expressions because transitional expressions vary for children with different positions. For example, when translating the last child of a *parallel* relation, we always use word "*and*" as the transitional expression seen in Figure 3, but we will not use it for the first child of a *parallel* relation. Therefore, in the training process we just keep the information of relationships and children's positions when converting

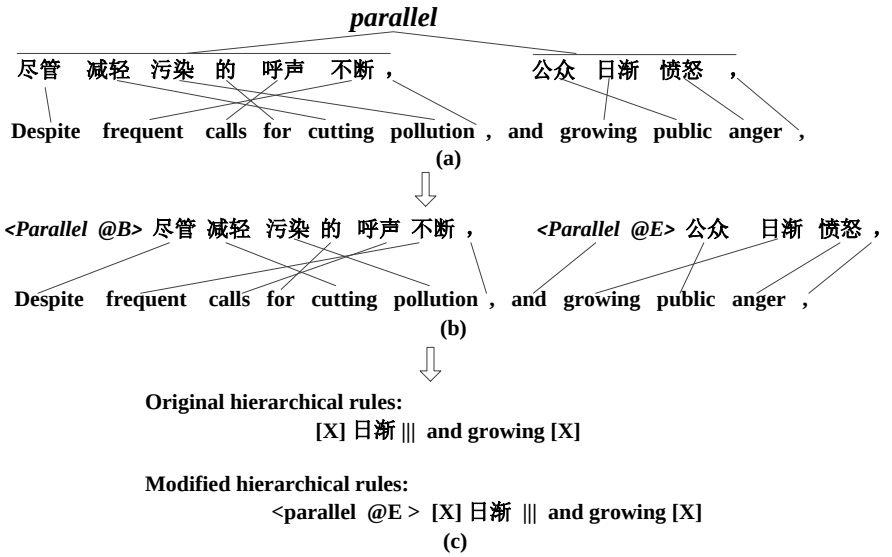


Figure 3: An example of modifying translation rules. @B means the current structure information comes from the first child, and @E means from the last child.

the source CSS to a tagged-flattened string.

Considering that the absolute position (index of the ***eu***, such as *1*, *2*, *3*) is somehow sparse in the corpus, we employ the relative position instead. *B* (*Beginning*) represents the first child of a relationship, *E* (*End*) means the last child of a relationship, and *M* (*Middle*) represents all the middle children.

Under this agreement, the original Chinese CSS-based tree will be converted to a new tagged-flattened string. Note the converting example from Figure 3(a) to Figure 3(b): node *parallel-[(1,1), (2,2)]* (see Figure 2) is converted to a flat string. Its first child is represented as <*parallel, @B*> with the semantic span, while the last child is <*parallel, @E*> with the corresponding semantic span.

We then perform word alignment on the modified bilingual sentences, and extract the new translation rules based on the new alignment, as shown in Figure 3(b) to Figure 3(c). Now the newly extracted rule "<*parallel, @E > [X] 日渐 ||| and growing [X]* " is tagged with cohesive information. Thus, if the similar relationship *parallel* occurs in the test source sentence, this type of rule is more likely to be chosen to generate the cohesive word "*and*" during decoding because it is more discriminating than the original rules (*[X] 日渐 ||| and growing [X]*). The conditional probabilities of the new translation rules are calculated following (Chiang, 2005).

3.2 CSS-based Transfer model

In general, according to formula (3), the translation quality based on the log-linear model is related tightly with the features chosen. Most translation systems adopt the features from a translation model, a language model, and sometimes a reordering model. To give a bonus to generating cohesive expressions during decoding, we have designed a special additional feature. The additional feature is represented as a probability calculated by a transfer model.

Given the source CSS information, we want our transfer model to predict the most possible cohesive expressions. For example, given two semantic spans with a *parallel* relationship and many translation candidates, our transfer model is expected to assign higher scores to those with transitional expressions such as "*and*" or "*as well as*".

Let $\boldsymbol{w} = w_0, w_1, ...w_n$ represent the transitional expressions observed in the target string. Our transfer model can be represented as a conditional probability:

$$P(\boldsymbol{w} \mid CSS) \quad (4)$$

By deriving each node of the CSS, we can obtain a factored formula:

$$P(\boldsymbol{w} \mid CSS) = \prod_{i,j} P(\boldsymbol{w}_{ij} \mid R_i, RP_j) \quad (5)$$

where $\boldsymbol{w}_{ij}$ is the transitional expression produced by the j^{th} child of the i^{th} node of the CSS. R_i is the relationship type of the i^{th} node. For the j^{th} child in the i^{th} node, RP_j is its relative position (*B, M* or *E*) introduced in Section 3.1.

The process of training this transfer model and smoothing is similar to the process of training a language model. We obtain the factored transfer probability as follows,

$$P(\boldsymbol{w}_{ij} \mid R_i, RP_j) = P(w_0 \mid R_i, RP_j) \prod_{k=1}^{n} P(w_k \mid w_0^{k-1}, R_i, RP_j) \quad (6)$$

where

$$\boldsymbol{w}_{ij} = w_0^{\,n} = w_0, ...w_n \quad (7)$$

Following (Bilmes and Kirchhoff, 2003), the conditional probabilities $P(w_k \mid w_0^{k-1}, R_i, RP_j)$ in formula (6) are estimated in the same way as a factored language model, which has the advantage of easily incorporating various linguistic information.

Considering that $\boldsymbol{w}_{ij}$ commonly appears at the beginning of the target translation of a source semantic span such as "*which …*", namely, the **left-frontier phrases,** we focus only on the left-frontier phrases when training this model. Note that if there exists a target word before a left frontier, and this word is aligned to *NULL*, we will expand the left frontier to this word. The expansion process will be repeated until there is no such word. For example, if we take the CSS and the alignment in Figure 3(a) for training, the left frontier of the second child will be expanded from "*growing*" to "*and*". In addition, taking the tri-gram left-frontier phrase for example, we can obtain a training sample such as $\boldsymbol{w}_{ij}$ = *and growing public*, *R=parallel*, *RP = E*.

By learning such probabilities for different transitional expressions conditioned on different relationships, we are able to capture the inner connection between the source CSS and the projected target cohesive phrases. Thus, during decoding, if we add the probability generated by the transfer model of $P(\boldsymbol{w} \mid CSS)$ as a feature in

formula (3), it will certainly contribute to selecting more cohesive candidates.

3.3 Elementary-Unit Cohesion Constraint

As mentioned in Section 3.2, in the transfer model, the transitional phrases are expected to occur at the left frontier of a projected span on target side. In fact, this depends on the assumption that the projected translations of any two disjoint source semantic spans are also disjoint to keep their own semantic integrity. We call this assumption the integrity assumption. This assumption is intuitive and supported by statistics. After analyzing 1,007 golden aligned Chinese-English sentence-pairs, we find that approximately 90% of the pairs comply with the assumption. However, in real automatically aligned noisy data, the ratio of complying pairs reduces to 71%[4]. Two projected translations that violate the integrity assumption may mutually overlap, which causes our confusion on where to extract the transitional phrases. In this case, extracted transitional phrases are likely to be wrong.

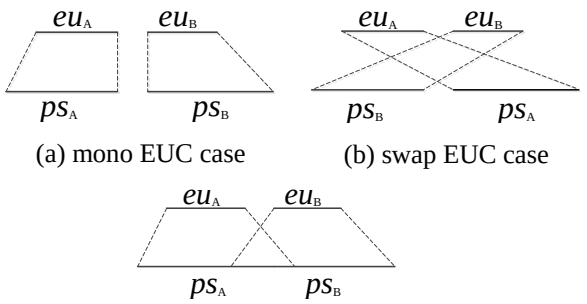
To increase the chance of extracting correct transitional phrases, the alignment results must be modified to reduce the impact of incorrect alignment. We propose a dynamic cleaning method to ensure that the most expressive transitional phrases fall in the accessible extraction range before training the transfer model.

3.3.1 EUC and non-EUC

As we have defined in Section 2, the minimal semantic span is called elementary unit (***eu***). If the source ***eu*** and its projected target span comply with the integrity assumption, we say that such an ***eu*** and its projected span have **Elementary-Unit-Cohesion (EUC).** We define EUC formally as follows.

Given two elementary units eu_A and eu_B, and their projected target spans ps_A and ps_B bound by the word alignment, the alignment complies with EUC only if there is no overlap between ps_A and ps_B. Otherwise, the alignment is called ***non-EUC***. The common ***EUC*** and ***non-EUC*** cases are illustrated in Figure 4.

EUC is the basic case for the integrity assumption. For the best cases, the elementary units comply with EUC, and thus the semantic spans combined by elementary units are certainly subject to the integrity assumption.



(a) mono EUC case (b) swap EUC case

(c) non-EUC case

Figure.4 The schematic diagram of *EUC* cases and *non-EUC* case.

3.3.2 A Dynamic Cleaning Method

An intuitive method to clean the alignment results is to drop off the noisy word-to-word links that cause non-EUC. Considering that the dropping process is a post-editing method for the original alignment obtained by a state-of-the-art aligner such as GIZA++, we do not expect over-deleting. Therefore, we tend to take a relatively conservative strategy to minimize the deleting operation.

Given a sentence-pair (***f***, ***e***), suppose that $\boldsymbol{f} = \{f_0, ..., f_i, ..., f_I\}$ is divided into M elementary units $U = \{u_0, ..., u_m, ..., u_M\}$, and ***e*** has N words, that is, $\boldsymbol{e} = \{e_0, ..., e_n, ..., e_N\}$. If A is the word alignment of (***f***, ***e***), then the goal is to construct the maximum subset $A^* \subseteq A$ under the condition that A^* is the word alignment with the constraint of EU. The search process can be described as the pseudo code in Figure 5.

In Figure 5, we scan each target word and each source ***eu*** to assign each word to a unique ***eu*** under the EUC constraint with the lowest cost. Function cost(n,m) in line 6 computes the counts of deleted links that force the n^{th} target word to align only to words in the range of the m^{th} ***eu***. For example, if the n^{th} target word is aligned to the i^{th}, $(i+1)^{th}$, and $(i+2)^{th}$ word in source side, while the i^{th} word belongs to u_{m_1} and the $(i+1)^{th}$ and $(i+2)^{th}$ words belong to u_{m_2}, then $\text{cost}(n, u_{m_1}) = 2$, and $\text{cost}(n, u_{m_2}) = 1$. In line 6, *Score*[$n$][$m$] saves a list of scores, each score computed by adding the current cost(n, m) with the history score of each list of *Score*[*n-1*].

[4] The aligning tool is GIZA++ with 5 iterations of Model 1, 5 iterations of HMM, and 10 iterations of Model 4. The GIZA++ code can be downloaded from https://code.google.com/p/giza-pp/

Before the next iteration, the bad branches are pruned, as seen in line 5. We adopt the following two ways to prune:

(1) EUC constraint: if the current link violates EUC alignment, delete it.

(2) Keep the hypothesis with a fixed maximum size to avoid too large a searching space.

```
//Pseudo code for dynamic cleaning
1: Score [N+1][M]={[0]}_{N×M}        /* initialize
                        cumulative cost score chart*/
2: Path [M]=[[]]              /*initialize tracking path*/
3: for n = 1 → N :{          /*  scan target words*/
4:   for  m = 0 → M −1:{      /*scan source U set */
5:     PrunePath();
          /* prune invalid  path and high-cost path*/
6:     Score[n][m]=GetScore(Score[n-1], cost(n, m))
     /*compute current cumulative cost score by previ-
                ous score and current cost*/
7:     SaveCurrentPath(Path[m]);
                    /*add current index to Path*/
8:  }//end m
9:}//end n
10: OptimalPath = arg max_{Path[m]} {Score[N][m]};
```

Figure 5. The pseudo code of dynamic cleaning method.

4 Experiments

4.1 Experimental Setup

To obtain the CSSs of Chinese sentences, we use the Chinese parser proposed in (Tu et al., 2013a). Their parser first segments the compound-complex sentence into a series of elementary units, and then builds structure of the hierarchical relationships among these elementary units. Their parser was reported to achieve an F-score for elementary unit segmentation of approximately 0.89. The *progressive*, *causal*, and *condition* terms of functional relationships can be recognized with precisions of 0.86, 0.8, and 0.75, respectively, while others, such as *purpose*, *parallel*, and *flowing*, achieve only 0.5, 0.59 and 0.62, respectively.

The translation experiments have been conducted in the Chinese-to-English direction. The bilingual training data for translation model and CSS-based transfer model is FBIS corpus with approximately 7.1 million Chinese words and 9.2 million English words. We obtain the word alignment with the grow-diag-final-and strategy with GIZA++. Before training the CSS-based transfer model, the alignment for transfer model is modified by our dynamic cleaning method. During the cleaning process, the maximum size of hypothesis is limited to 5. A 5-gram language model is trained with SRILM[5] on the combination of the Xinhua portion of the English Gigaword corpus combined with the English part of FBIS. For tuning and testing, we use NIST03 evaluation data as the development set. NIST04/05/06, CWMT08-Development [6] and CWMT08-Evaluation data are used for testing under the measure metric of BLEU-4 (Papineni et al. 2002) with the shortest length penalty.

Table 1 shows how the CSS is distributed in all testing sets. According to the statistics in Table 1, we see that CSS is really widely distributed in the NIST and CWMT corpora, which implies that the translation quality may benefit substantially from the CSS information, if it is well considered in SMT.

	Total	CSS	Ratio(%)
NIST04	1,788	1,307	73.1
NIST05	1,082	849	78.5
NIST06	1,000	745	74.5
CWMT08-Dev.	1,006	818	81.3
CWMT08-Eval.	1,006	818	81.3

Table 1. The numbers of sentences and the CSS ratios of all sentences. CWMT08-Dev. is short for CWMT08 Development data and CWMT08-Eval. is CWMT08 Evaluation data.

4.2 Extracted Transitional Expressions

Eleven types of Chinese functional relationships and their English left-frontier phrases (tri-gram) learned by our transfer model are given in Table 2.

The results in Table 2 show that some left-frontier phrases reflect the source functional relationship well, especially for those with better precision of relationship recognition, such as *progressive, causal* and *condition*. Conversely, lower precision of relationship recognition may weaken the learning ability of the transfer model. For example, noisy left-frontier phrases are easily generated under relationships such as *parallel* and *purpose*.

[5] http://www.speech.sri.com/projects/srilm/

[6] The China Workshop on Machine Translation

Relation	Left-frontier phrases (tri-gram)
parallel	as well as; at the same; …
progressive	but will also; in addition to;…
causal	therefore , the; for this reason; as a result; because it is; so it is;…
condition	as long as; only when the…
hypothesis	if we do; if it is; if the us; …
alternative	regardless of whether;…
purpose	it is necessary; further promote the ;…
explanation	that is ,; the first is; first is the;…
adversative	however , the ; but it is; …
flowing	this is a; which is an; …
consequence	so that the; to ensure that…

Table 2. Chinese functional relations and their corresponding English left-frontier phrases learned by our transfer model. The noun phrases starting with a definite / indefinite word are filtered because they are unlikely to be the transitional phrases.

4.3 Results on SMT with Different Strategies

For this work, we use an in-house decoder to build the SMT baseline; it combines the hierarchical phrase-based translation model (Chiang, 2005; Chiang, 2007) with the BTG (Wu, 1996) reordering model (Xiong et al., 2006; Zens and Ney, 2006; He et al., 2010).

To test the effectiveness of the proposed models, we have compared the translation quality of different integration strategies. First, we adopted only the tagged-flattened rules in the hierarchical translation system. Next, we added the log probability generated by the transfer model as a feature into the baseline features. The baseline features include bi-directional phrase translation probabilities, bi-directional lexical translation probabilities, the BTG re-ordering features, and the language model feature. The tri-gram left-frontier phrase was adopted in the experiment. Then the probability generated by the transfer model with EUC constraint is added. Finally, we incorporated the tagged-flattened rules and the additional transfer model feature together.

Table 3 shows the results of these different integrated strategies. In Table 3, almost all BLEU scores are improved, no matter what strategy is used. In particular, the best performance marked in bold is as high as 1.24, 0.94, and 0.82 BLEU points, respectively, over the baseline system on NIST04, CWMT08 Development, and CWMT08 Evaluation data. The strategy of "**TFS+ Flattened Rule**" is the most stable. Meanwhile the "**Flattened Rule**" achieves better performance than "**TFS**". The merits of "**Flattened Rule**" are two-fold: 1) In training process, the new word alignment upon modified sentence pairs can align transitional expressions to flattened CSS tags; 2) In decoding process, the CSS-based rules are more discriminating than the original rules, which is more flexible than "TFS". From the table, we cannot conclude that the EUC constraint will certainly promote translation quality, but the transfer model performs better with the constraint on most testing sets.

4.4 Analysis of Different Effects of Different N-grams

As mentioned in Section 4.3, we have noted the effectiveness of tri-gram transfer model, which means $n = 2$ in formula (7). In fact, the lengths of common transitional expressions vary from one word to several words. To evaluate the effects of different n-grams for our proposed transfer model, we compared the uni-/bi-/tri-gram transfer models in SMT, and illustrate the results in Fig-

	NIST04	NIST05	NIST06	CWMT08's Dev.	CWMT08's Eval.
Baseline	33.42	31.99	33.88	26.14	23.88
+Flattened Rule	34.54**	32.32	**34.58****	26.79**	**24.70****
+TFS (without EUC)	33.93**	32.04	34.40*	26.44	24.58**
+TFS	33.84**	**32.63***	34.15	**27.08****	24.65**
+TFS+ Flattened Rule	**34.66****	32.54	34.52**	26.87**	24.49**

+ **Flattened Rule**: only use the tagged-flattened translation rules
+ **TFS:** only use the transfer model score as an additional feature (based on 3-gramtransitional phrase)
+ **TFS** + **Flattened Rule**: both are used
*: value with * means that it is significantly better than the baseline with p<0.05
**: value with ** means that it is significantly better than the baseline with p<0.01

Table 3. BLEU scores of the testing sets with different integrating strategies

ure 6. In this experiment, the CSS-based translation rules and the CSS-based transfer model are both incorporated. Considering time and computing resources, in the rest of our paper, our analysis is conducted on NIST05 and NIST06.

We choose $n = 0, 1, 2$ in this experiment for that the common English transitional expressions are primarily conjunctions, most of which are less than 4 words. Results in Figure 6 show that the uni-gram and tri-gram transitional expressions seem more fitting for our transfer model. One possible reason is that uni-gram or tri-gram conjunctions are more utilized in an English text. In a conjunction expression list proposed by (Williams, 1983) which summarizes the different kinds of conjunctions based on the work of Halliday and Hassan (1976), we obtain the statistical results on uni-/bi-/tri-gram expressions, which are about 52.1%/16.9%/23.9% respectively.

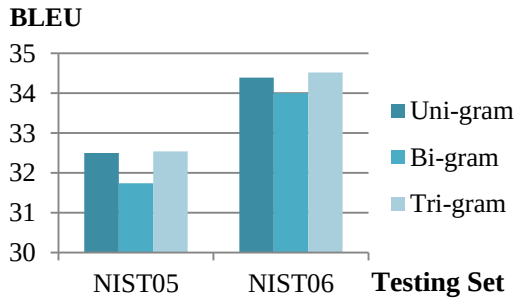


Figure 6. Different translation qualities along with different n-grams for transfer model.

4.5 Experiments on Big Training Data

To further evaluate the effectiveness of the proposed models, we also conducted an experiment on a larger set of bilingual training data from the LDC corpus[7] for translation model and transfer model. The training corpus contains 2.1M sentence pairs with approximately 27.7M Chinese words and 31.9M English words. All the other settings were the same as the SMT experiments of sub-section 4.3. The final BLEU scores on NIST05 and NIST06 are given in Table 4.

The results in Table 4 further verify the effectiveness of our proposed models. The best performance with bold marking scored as high as 0.83 and 0.64 BLEU points, respectively over the baseline system on NIST05 and NIST06 evaluation data.

	NIST05	NIST06
Baseline	35.20	35.52
+Flattened Rule	**36.03****	36.10*****
+TFS	35.56*	36.04*
+TFS +Flattened Rule	36.02**	36.16**

+ **Flattened Rule**: only use the tagged-flattened translation rules
+ **TFS:** only use the transfer model score as an additional feature (3-gram transitional phrase)
+ **TFS** + **Flattened Rule**: both are used
*: value with * means that it is significantly better than the baseline with $p<0.05$
**: value with ** means that it is significantly better than the baseline with $p<0.01$

Table 4. BLEU scores on the large-scale training data.

4.6 Translation Examples

Two SMT examples of Chinese-to-English are given in Table 5. We observe that compared to the baseline, our approach has obvious advantages on translating the implicit relations, due to generating translational expressions on target side. Moreover, with the transitional expressions, cohesion of the entire translation improves. Notably, the transitional expressions in this work like "*including*, *there are*, *the core of which*" are not linguistic conjunctions. We would like to call them "generalized" conjunctions, because they tie semantic fragments together, analogously to linguistic conjunctions.

5 Related Work

Improving cohesion for complex sentences or discourse translation has attracted much attention in recent years. Such research efforts can be roughly divided into two groups: 1) research on lexical cohesion, which mainly contributes to the selection of generated target words; 2) efforts to improve the grammatical cohesion, such as disambiguation of references and connectives.

In lexical cohesion work, (Gong et al., 2011; Xiao et al., 2011; Wong and Kit, 2012) built discourse-based models to ensure lexical cohesion or consistency. In (Xiong et al., 2013a), three different features were designed to capture the lexical cohesion for document-level machine translation. (Xiong et al., 2013b) incorporated lexical-chain-based models (Morris and Hirst, 1991) into machine translation. They generated the target lexical chains based on the source

[7] LDC category number: LDC2000T50, DC2002E18, LDC2003E07, LDC2004T07, LDC2005T06, LDC2002L27, LDC2005T10 and LDC2005T34.

Source	过去三年中，已有三对染色体完成排序， 包括第二十对、第二十一对和第二十二 对 。
Reference	In the past three years, the sequencing of three chromosomes has been completed, including chromosomes 20 , 21 , and 22 .
Baseline	In the past three years , now has three terms of the completion of the chromosomes , 20 , 21 and 22 .
Improved	In the past three years , **there are** three chromosomes to accomplish , **including** 20 , 21 and 22 .
Source	上述主张构成了一个中国原则的基本涵义，核心是维护中国的主权和领土完整。
Reference	The above-mentioned propositions constitute the basic connotation of this one-china principle with safeguarding china ' s sovereignty and territorial integrity as its core .
Baseline	The above-mentioned propositions constitute the basic meaning of the one-china principle is the core of safeguard china ' s sovereignty and territorial integrity .
Improved	The above-mentioned propositions constitute the basic meaning of the one-china principle , **the core of which** is to safeguard china ' s sovereignty and territorial integrity .

Table 5. Examples of baseline and the improved system outputs.

chains via maximum entropy classifiers, and used the target chains to work on the word selection.

Limited work has been conducted on grammatical cohesion. (Marcu et al., 2000) designed a discourse structure transfer module, but it focused on converting the semantic structure rather than actual translation. (Tu et al., 2013b) provided a Rhetorical-Structure-Theory-based tree-to-string translation method for complex sentences with explicit relations inspired by (Marcu et al., 2000), but their models worked only for explicit functional relations, and they were concerned mainly with the translation integrity of semantic span rather than cohesion. (Meyer and Popescu-Belis, 2012) used sense-labeled discourse connectives for machine translation from English to French. They added the labels assigned to connectives as an additional input to an SMT system, but their experimental results show that the improvements under the evaluation metric of BLEU were not significant. (Nagard and Koehn, 2010) addresses the problems of reference or anaphora resolution inspired by work of Mitkov et al. (1995).

To the best of our knowledge, our work is the first attempt to exploit the source functional relationship to generate the target transitional expressions for grammatical cohesion, and we have successfully incorporated the proposed models into an SMT system with significant improvement of BLEU metrics.

6 Conclusion

In this paper, we focus on capturing cohesion information to enhance the grammatical cohesion of machine translation. By taking the source CSS into consideration, we build bridges to connect the source functional relationships in CSS to target transitional expressions; such a process is very similar to human translating.

Our contributions can be summarized as: 1) the new translation rules are more discriminative and sensitive to cohesive information by converting the source string into a CSS-based tagged-flattened string; 2) the new additional features embedded in the log-linear model can encourage the decoder to produce transitional expressions. The experimental results show that significant improvements have been achieved on various test data, meanwhile the translations are more cohesive and smooth, which together demonstrate the effectiveness of our proposed models.

In the future, we will extend our methods to other translation models, such as the syntax-based model, to study how to further improve the performance of SMT systems. Besides, more language pairs with various linguistic structures will be taken into consideration.

Acknowledgement

We would like to thank Jiajun Zhang for providing the BTG-based hierarchical decoder. The research work has been partially funded by the Natural Science Foundation of China under Grant No. 61333018, the Hi-Tech Research and Development Program (“863” Program) of China under Grant No. 2012AA011101, and also the Key Project of Knowledge Innovation Program of Chinese Academy of Sciences under Grant No. KGZD-EW-501 as well.

References

Jeff A. Bilmes and Katrin Kirchhoff. *Factored language models and generalized parallel backoff*. In Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003--short papers-Volume 2: 4-6.

David Chiang. 2005. *A hierarchical phrase-based model for statistical machine translation*. In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics, pages 263–270.

David Chiang. 2007. *Hierarchical phrase-based translation*. Computational Linguistics, pages 33(2):201–228.

Zhengxian Gong, Min Zhang, and Guodong Zhou. *Cache-based document-level statistical machine translation*, 2011, Edinburgh, Scotland, UK. In Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 909–919.

Liane Guillou. 2013. *Analysing lexical consistency in translation*. In Proceedings of the Workshop on Discourse in Machine Translation, pages 10–18, Sofia

Michael A.K. Halliday, Hasan R. *Cohesion in English*. 1976. London: Longman.

Zhongjun He, Yao Meng, and Hao Yu. 2010b. *Maximum Entropy Based Phrase Reordering for Hierarchical Phrase-based Translation*. In Proc. of the Conf. on Empirical Methods for Natural Language Processing (EMNLP), pages 555–563.

Annie Louis and Ani Nenkova. 2012. *A coherence model based on syntactic patterns*. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 1157–1168, Jeju Island, Korea, July.

William C Mann and Sandra A Thompson. 1988. *Rhetorical structure theory: Toward a functional theory of text organization*. Text, 8(3):243–281.

Ruslan Mitkov, Sung-Kwon Choi, and Randall Sharp. 1995. *Anaphora resolution in Machine Translation*. In Proceedings of the Sixth International Conference on Theoretical and Methodological Issues in Machine Translation.

Thomas Meyer and Andrei Popescu-Belis. *Using sense-labeled discourse connectives for statistical machine translation*, 2012, In Proceedings of the Joint Workshop on Exploiting Synergies between Information Retrieval and Machine Translation (ESIRMT) and Hybrid Approaches to Machine Translation (HyTra), pages:129-138.

Jane Morris and Graeme Hirst. 1991. *Lexical cohesion computed by thesaural relations as an indicator of the structure of text*. Comput. Linguist., 17(1):21–48, March.

Ronan L Nagard and Philipp Koehn. 2010, *Aiding pronoun translation with co-reference resolution*, In proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR, pages 252-261.

Franz J Och and Hermann Ney. 2002. *Discriminative training and maximum entropy models for statistical machine translation*. In Proc. of ACL, pages 295–302.

Kishore Papineni, Salim Roukos, Todd Ward, et al. 2002, *BLEU: a method for automatic evaluation of machine translation*. In proceedings of the 40th annual meeting on association for computational linguistics. pages: 311-318.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. *The Penn Discourse Treebank 2.0*. In Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008).

Williams Ray. *Teaching the Recognition of Cohesive Ties in Reading a Foreign*, 1983. Reading in a foreign language, 1(1), pages: 35-52.

Radu Soricut and Daniel Marcu. 2003. *Sentence level discourse parsing using syntactic and lexical information*. In Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1, pages 149–156.

Mei Tu, Yu Zhou, and Chengqing Zong. 2013a, *A Novel Translation Framework Based on Rhetorical Structure Theory*. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, short paper, Sofia, Bulgaria, pages 370–374.

Mei Tu, Yu Zhou, Chengqing Zong. 2013b, *Automatically Parsing Chinese Discourse Based on Maximum Entropy*. In The 2nd Conference on Natural Language Processing & Chinese Computing.

Ashish Vaswani, Liang Huang and David Chiang, Huang L, Chiang D. 2012, *Smaller alignment models for better translations: unsupervised word alignment with the l 0-norm*. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1,pages 311-319.

Tong Xiao, Jingbo Zhu, Shujie Yao, and Hao Zhang. *Document-level consistency verification in machine translation*. September 2011, Xiamen, China. In Proceedings of the 2011 MT summit XIII, pages 131–138.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. *Maximum entropy based phrase reordering model for statistical machine translation*. In Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics, pages 521–528.

Deyi Xiong, Guosheng Ben, Min Zhang, Yajuan Lv, and Qun Liu. 2013 (a). *Modeling lexical cohesion for document-level machine translation*. In Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI-13), Beijing, China, August.

Deyi Xiong, Ding Yang, Min Zhang and Chew Lim Tan, 2013 (b). *Lexical Chain Based Cohesion Models for Document-Level Statistical Machine Translation*. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages: 1563-1573.

Richard Zens and Hermann Ney. 2006. *Discriminative reordering models for statistical machine translation*. In Proceedings of theWorkshop on Statistical Machine Translation, pages 55–63.

Qiang Zhou, 2004, *Annotation Scheme for Chinese Treebank*, Journal of Chinese Information Processing, 18(4): 1-8.

Adaptive HTER Estimation for Document-Specific MT Post-Editing

Fei Huang *
Facebook Inc.
Menlo Park, CA
feihuang@fb.com

Jian-Ming Xu **Abraham Ittycheriah** **Salim Roukos**
IBM T.J. Watson Research Center
Yorktown Heights, NY
{jianxu, abei, roukos}@us.ibm.com

Abstract

We present an adaptive translation quality estimation (QE) method to predict the human-targeted translation error rate (HTER) for a document-specific machine translation model. We first introduce features derived internal to the translation decoding process as well as externally from the source sentence analysis. We show the effectiveness of such features in both classification and regression of MT quality. By dynamically training the QE model for the document-specific MT model, we are able to achieve consistency and prediction quality across multiple documents, demonstrated by the higher correlation coefficient and F-scores in finding *Good* sentences. Additionally, the proposed method is applied to IBM English-to-Japanese MT post editing field study and we observe strong correlation with human preference, with a 10% increase in human translators' productivity.

1 Introduction

Machine translation (MT) systems suffer from an inconsistent and unstable translation quality. Depending on the difficulty of the input sentences (sentence length, OOV words, complex sentence structures and the coverage of the MT system's training data), some translation outputs can be perfect, while others are ungrammatical, missing important words or even totally garbled. As a result, users do not know whether they can trust the translation output unless they spend time to analyze the MT output. This shortcoming is one of the main obstacles for the adoption of MT systems, especially in machine assisted human translation: MT post-editing, where human translators have an option to edit MT proposals or translate from scratch. It has been observed that human translators often discard MT proposals even if some are very accurate. If MT proposals are used properly, post-editing can increase translators productivity and lead to significant cost savings. Therefore, it is beneficial to provide MT confidence estimation, to help the translators to decide whether to accept MT proposals, making minor modifications on MT proposals when the quality is high or translating from scratching when the quality is low. This will save the time of reading and parsing low quality MT and improve user experience.

In this paper we propose an adaptive quality estimation that predicts sentence-level human-targeted translation error rate (HTER) (Snover et al., 2006) for a document-specific MT post-editing system. HTER is an ideal quality measurement for MT post editing since the reference is obtained from human correction of the MT output. Document-specific MT model is an MT model that is specifically built for the given input document. It is demonstrated in (Roukos et al., 2012) that document-specific MT models significantly improve the translation quality. However, this raises two issues for quality estimation. First, existing approaches to MT quality estimation rely on lexical and syntactical features defined over parallel sentence pairs, which includes source sentences, MT outputs and references, and translation models (Blatz et al., 2004; Ueffing and Ney, 2007; Specia et al., 2009a; Xiong et al., 2010; Soricut and Echihabi, 2010a; Bach et al., 2011). Therefore, when the MT quality estimation model is trained,

*This work was done when the author was with IBM Research.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 861–870,
Baltimore, Maryland, USA, June 23-25 2014.

it can not be adapted to provide accurate estimates on the outputs of document-specific MT models. Second, the MT quality estimation might be inconsistent across different document-specific MT models, thus the confidence score is unreliable and not very helpful to users.

In contrast to traditional static MT quality estimation methods, our approach not only trains the MT quality estimator dynamically for each document-specific MT model to obtain higher prediction accuracy, but also achieves consistency over different document-specific MT models. The experiments show that our MT quality estimation is highly correlated with human judgment and helps translators to increase the MT proposal adoption rate in post-editing.

We will review related work on MT quality estimation in section 2. In section 3 we will introduce the document-specific MT system built for post-editing. We describe the static quality estimation method in section 4, and propose the adaptive quality estimation method in section 5. In section 6 we demonstrate the improvement of MT quality estimation with our method, followed by discussion and conclusion in section 7.

2 Related Work

There has been a long history of study in confidence estimation of machine translation. The work of (Blatz et al., 2004) is among the best known study of sentence and word level features for translation error prediction. Along this line of research, improvements can be obtained by incorporating more features as shown in (Quirk, 2004; Sanchis et al., 2007; Raybaud et al., 2009; Specia et al., 2009b). Soricut and Echihabi (2010b) proposed various regression models to predict the expected BLEU score of a given sentence translation hypothesis. Ueffing and Hey (2007) introduced word posterior probabilities (WPP) features and applied them in the n-best list reranking. Target part-of-speech and null dependency link are exploited in a MaxEnt classifier to improve the MT quality estimation (Xiong et al., 2010).

Quality estimation focusing on MT post-editing has been an active research topic, especially after the WMT 2012 (Callison-Burch et al., 2012) and WMT2013 (Bojar et al., 2013) workshops with the "Quality Estimation" shared task. Biçici et al. (2013) proposes a number of features measuring the similarity of the source sentence to the source side of the MT training corpus, which, combined with features from translation output, achieved significantly superior performance in the MT QE evaluation. Felice and Specia (2012) investigates the impact of a large set of linguistically inspired features on quality estimation accuracy, which are not able to outperform the shallower features based on word statistics. González-Rubio et al. (2013) proposed a principled method for performing regression for quality estimation using dimensionality reduction techniques based on partial least squares regression. Given the feature redundancy in MT QE, their approach is able to improve prediction accuracy while significantly reducing the size of the feature sets.

3 Document-specific MT System

In our MT post-editing setup, we are given documents in the domain of software manuals, technical outlook or customer support materials. Each translation request comes as a document with several thousand sentences, focusing on a specific topic, such as the user manual of some software.

The input documents are automatically segmented into sentences, which are also called *segments*. Thus in the rest of the paper we will use sentences and segments interchangeably. Our parallel corpora includes tens of millions of sentence pairs covering a wide range of topics. Building a general MT system using all the parallel data not only produces a huge translation model (unless with very aggressive pruning), the performance on the given input document is suboptimal due to the unwanted dominance of out-of-domain data. Past research suggests using weighted sentences or corpora for domain adaptation (Lu et al., 2007; Matsoukas et al., 2009; Foster et al., 2010). Here we adopt the same strategy, building a document-specific translation model for each input document.

The document-specific system is built based on sub-sampling: from the parallel corpora we select sentence pairs that are the most similar to the sentences from the input document, then build the MT system with the sub-sampled sentence pairs. The *similarity* is defined as the number of n-grams that appear in both source sentences, divided by the input sentence's length, with higher weights assigned to longer n-grams. From the extracted sentence pairs, we utilize the standard pipeline in SMT system building: word align-

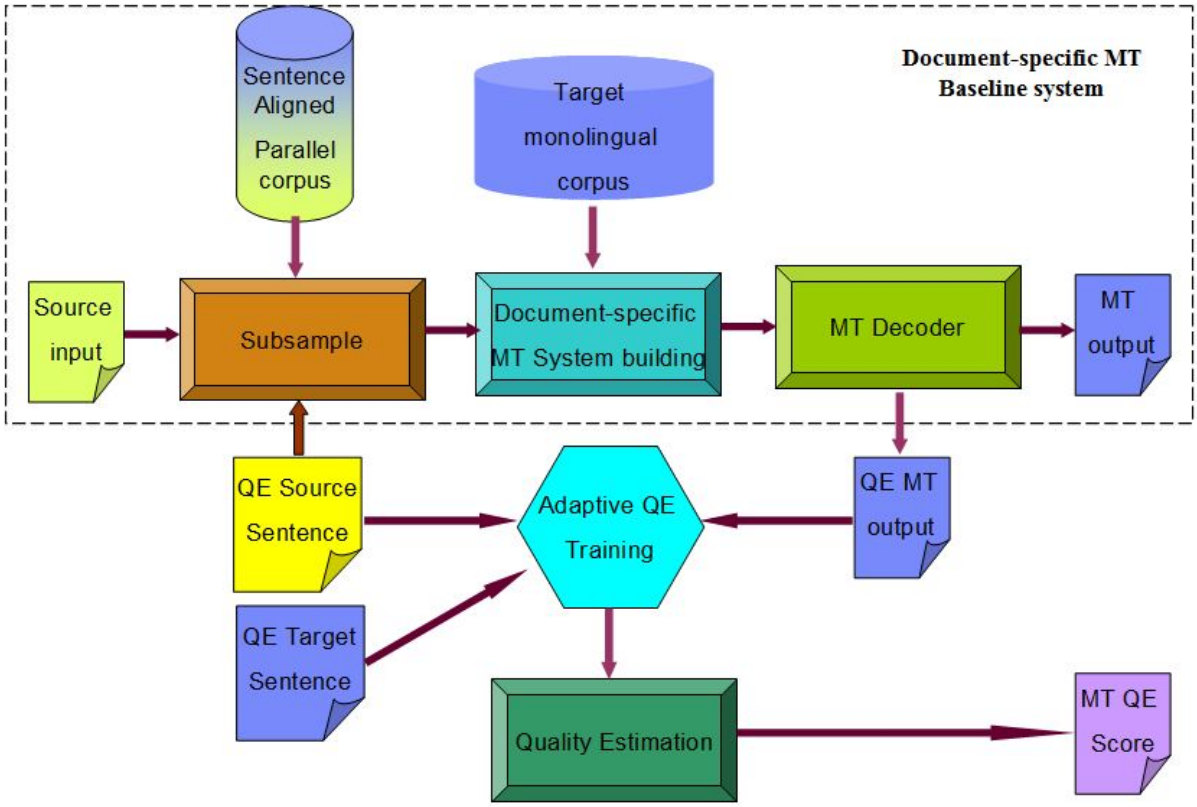


Figure 1: Adaptive QE for document-specific MT system.

ment (HMM (Vogel et al., 1996) and MaxEnt (Ittycheriah and Roukos, 2005) alignment models, phrase pair extraction, MT model training (Ittycheriah and Roukos, 2007) and LM model training. The top region within the dashed line in Figure 1 shows the overall system built pipeline.

3.1 MT Decoder

The MT decoder (Ittycheriah and Roukos, 2007) employed in our study extracts various features (source words, morphemes and POS tags, target words and POS tags, etc.) with their weights trained in a maximum entropy framework. These features are combined with other features used in a typical phrase-based translation system. Altogether the decoder incorporates 17 features with weights estimated by PRO (Hopkins and May, 2011) in the decoding process, and achieves state-of-the-art translation performance in various Arabic-English translation evaluations (NIST MT2008, GALE and BOLT projects).

4 Static MT Quality Estimation

MT quality estimation is typically formulated as a prediction problem: estimating the confidence score or translation error rate of the translated sentences or documents based on a set of features. In this work, we adopt HTER in (Snover et al., 2006) as our prediction output. HTER measures the percentage of insertions, deletions, substitutions and shifts needed to correct the MT outputs. In the rest of the paper, we use TER and HTER interchangably.

In this section we will first introduce the set of features, and then discuss MT QE problem from classification and regression point of views.

4.1 Features for MT QE

The features for quality estimation should reflect the complexity of the source sentence and the decoding process. Therefore we conduct syntactic analysis on the source sentences, extract features from the decoding process and select the following 26 features:

- 17 decoding features, including phrase translation probabilities (source-to-target and target-to-source), word translation probabilities (also in both directions), maxent probabilities[1], word count, phrase count, distor-

[1]The maxent probability is the translation probability

tion probabilities, as well as a set of language model scores.

- Sentence length, i.e., the number of words in the source sentence.
- Source sentence syntactic features, including the number of *noun phrases, verb phrases, adjective phrases, adverb phrases*, as inspired by (Green et al., 2013).
- The length of verb phrases, because verbs are typically the roots in dependency structure and they have more varieties during translation.
- The maximum length of source phrases in the final translation, since longer matching source phrase indicates better coverage of the input sentence with possibly better translations.
- The number of phrase pairs with high *fuzzy match (FM)* score. The high FM phrases are selected from sentence pairs which are closest in terms of n-gram overlap to the input sentence. These sentences are often found in previous translations of the software manual, and thus are very helpful for translating the current sentence.
- The average translation probability of the phrase translation pairs in the final translation, which provides the overall translation quality on the phrase level.

The first 17 features come from the decoding process, which are called "decoding features". The remaining 9 features not related to the decoder are called "external features". To evaluate the effectiveness of the proposed features, we train various classifiers with different feature configurations to predict whether a translation output is useful (with lower TER) as described in the following section.

4.2 MT QE as Classification

Predicting TER with various input features can be treated as a regression problem. However for the post-editing task, we argue that it could also be cast as a classification problem: MT system users (including the translators) are often interested to know whether a given translation is reasonably good or not. If useful, they can quickly look through the translation and make minor modifications. On the other hand, they will just skip reading and parsing the bad translation, and prefer to translate by themselves from scratch. Therefore we also develop algorithms that classify the translation at different levels, depending on whether the TER is less than a given threshold. In our experiments, we set TER=0.1 as the threshold.

Configuration	Training set	Test set
Baseline (All negative)	80%	77%
17 decoding features only	89%	79%
9 external features only	85%	81%
total 26 features	92%	83%

Table 1: QE classification accuracy with different feature configurations

We randomly select one input document with 2067 sentences for the experiment. We build a document-specific MT system to translate this document, then ask human translator to correct the translation output. We compute TER for each sentence using the human correction as the reference. The TER of the whole document is 0.31, which means about 30% errors should be corrected. In the classification task, our goal is to predict whether a sentence is a *Good* translation (with TER $\leq$ 0.1), and label them for human correction. We adopt a decision tree-based classifier, experimenting with different feature configurations. We select the top 1867 sentences for training and the bottom 200 sentences for test. In the test set, there are 46 sentences with TER $\leq$ 0.1. Table 1 shows the classification accuracy.

First we can see that as the overall TER is around 0.3, predicting all the sentences being *negative* already has a strong baseline: 77%. However this is not helpful for the human translators, because that means they have to translate every sentence from scratch, and consequently there is no productivity gain from MT post-editing. If we only use the 17 decoding features, it improves the classification accuracy by 9% on the training set, but only 2% on the test set. This is probably due to the overfitting when training the decision tree classifier. While using the 7 external features, the gain on training set is less but the gain on the test set

derived from a Maximum Entropy translation model (Ittycheriah and Roukos, 2005).

is greater (4% improvement), because the translation output is generated based on the log-linear combination of these decoding features, which are biased towards the final translations. The external features capture the syntactic structure of the source sentence, as well as the coverage of the training data with regard to the input sentence, which are good indicators of the translation quality. Combining both the decoding features and the external features, we observed the best accuracy on both the training and test set. We will use the combined 26 features in the following work.

4.3 MT QE as Regression

For the QE regression task, we predict the TER for each sentence translation using the above 26 features. We experiment with several classifiers: linear regression model, decision tree based regression model and SVM model. With the same training and test data set up, we predict the TER for each sentence in the test set, and compute the correlation coefficient (r) and root mean square error (RMSE). Our experiments show that the decision tree-based regression model obtains the highest correlation coefficients (0.53) and lowest RMSE (0.23) in both the training and test sets. We will use this model for the adaptive MT QE in the following work.

5 Adaptive MT Quality Estimation

The above QE regression model is trained on a portion of the sentences from the input document, and evaluated on the remaining sentences from the same document. One would like to know whether the trained model can achieve consistent TER prediction accuracy on other documents. When we use the cross-document models for prediction, the correlation is significantly worse (the details are discussed in section 6.1). Therefore it is necessary to build a QE regression model that's robust to different document-specific translation models. To deal with this problem, we propose this adaptive MT QE method described below.

Our proposed method is as follows: we select a fixed set of sentence pairs (S_q, R_q) to train the QE model. The source side of the QE training data S_q is combined with the input document S_d for MT system training data subsampling. Once the document-specific MT system is trained, we use it to translate both the input document and the source QE training data, obtaining the translation T_d and

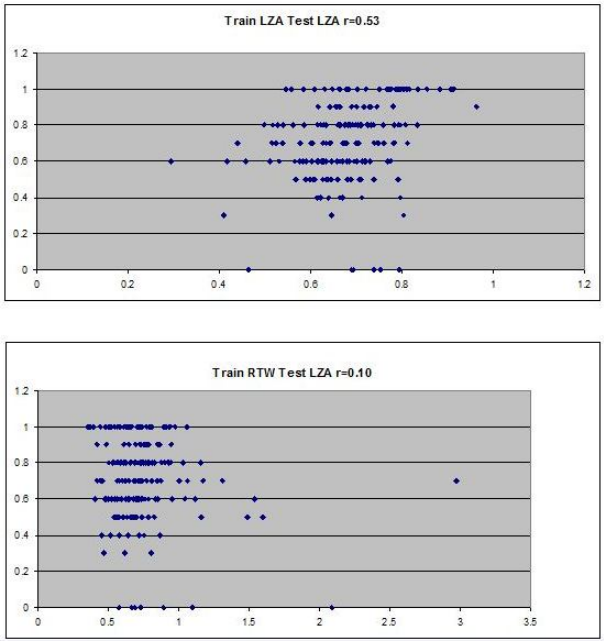


Figure 2: Correlation coefficient r between predicted TER (x-axis) and true TER (y-axis) for QE models trained from the same document (top figure) or different document (bottom figure).

T_q . We compute the TER of T_q using R_q as the reference, and train a QE regression model with the 26 features proposed in section 4.1. Then we use this *document-specific* QE model to predict the TER of the document translation T_d. As the QE model is adaptively re-trained for each document-specific MT system, its prediction is more accurate and consistent. Figure 1 shows the flow of our MT system with the adaptive QE training integrated as part of the built.

6 Experiments

In this section, we first discuss experiments that compare adaptive QE method and static QE method on a few documents, and then present results we obtained after deploying the adaptive QE method in an English-to-Japanese MT Post-Editing project. As mentioned before, the main motivation for us to develop MT QE classification scheme is that translators often discard good MT proposals and translate the segments from scratch. We would like to provide translators with some guidance on reasonably good MT proposals–the sentences with low TERs–to help them increase the leverage on MT proposals to achieve improved productivity.

6.1 Evaluation on Test Set

Our experiment and evaluation is conducted over three documents, each with about 2000 segments. We first build document-specific MT model for each document, then ask human translators to correct the MT outputs and obtain the reference translation. In a typical MT QE scenario, the QE model is pre-trained and applied to various MT outputs, even though the QE training data and MT outputs are generated from different translation models. To evaluate whether such model mismatch matters, we compare the cross-model QE with the same-model QE, where the QE training data and the MT outputs are generated from the same MT model.

We select one document LZA with 2067 sentences. We use the first 1867 sentences to train the static QE model and the remaining 200 sentences are used as test set for TER prediction. We compute the correlation coefficient (r) between each predicted TER and true TER, as shown in Figure 2. We find that the TER predictions are reasonably correct when the training and test sentences are from the same MT model (the top figure), with correlation coefficients around 0.5. For the cross-model QE, we train a static QE model with 1867 sentences from another document RTW, and use it to predict the TER of the same 200 sentences from document LZA (the bottom figure). We observe significant degradation of correlation coefficient, dropping from 0.5 to 0.1. This degradation and unstable nature is the prime motivation to develop a more robust MT quality estimation model.

We select 1700 sentences from multiple previously translated documents as the QE training data, which are independent of the test documents. We train the static QE model with this training set, including the source sentences, references and MT outputs (from multiple translation models). To train the adaptive QE model for each test document, we build a translation model whose subsampling data includes source sentences from both the test document and the QE training data. We translate the QE source sentences with this newly built MT model, and the translation output is used to train the QE model specific to each test document. We compare these two QE models on three documents, LZA, RTW and WC7, measuring r and RMSE for each QE model. The result is shown in Table 2. We find that the adaptive QE model demonstrates higher r and lower RMSE than the static QE model for all the test documents.

Besides the general correlation with human judgment, we particularly focus on those reasonably good translations, i.e., the sentences with low TERs which can help improve the translator's productivity most. Here we report the precision, recall and F-score of finding such "*Good*" sentences (with TER ≤ 0.1) on the three documents in Table 3. Again, the adaptive QE model produces higher recall, mostly higher precision, and significantly improved F-score. The overall F-score of the adaptive QE model is 0.28[2]. Compared with the static QE model's 0.17 F-score, this is relatively 64% improvement.

In the adaptive QE model, the source side QE training data is included in the subsampling process to build the document-specific MT model. It would be interesting to know whether this process will negatively affect the MT quality. We evaluate the TER of MT outputs with and without the adaptive QE training on the same three documents. As seen in Table 4, we do not notice translation quality degradation. Instead, we observe slightly improvement on two document, with TERs reduction by 0.1-0.4 pt. As our MT model training data include proprietary data, the MT performance is significantly better than publicly available MT software.

6.2 Impact on Human Translators

We apply the proposed adaptive QE model to large scale English-to-Japanese MT Post-Editing project on 36 documents with 562K words. Each English sentence can be categorized into 3 classes:

- Exact Match (**EM**): the source sentence is completely covered in the bilingual training corpora thus the corresponding target sentence is returned as the translation;
- Fuzzy Match (**FM**): the source sentence is similar to some sentence in the training data (similarity measured by string editing distance), the corresponding fuzzy match target sentence (FM proposal) as well as the MT translation output (MT proposal) are returned for human translators to select and correct;
- No Proposal (**NP**): there is no close match source sentences in the training data (the FM

[2]The adaptive QE model obtains much higher F-score (80%) on the rest of the sentences (with TER > 0.1).

Document	LZA		RTW		WC7	
Num. of Sents	2067		2003		2405	
	$r \uparrow$	RMSE $\downarrow$	$r \uparrow$	RMSE $\downarrow$	$r \uparrow$	RMSE $\downarrow$
Static QE	0.10	0.38	0.40	0.32	0.13	0.36
Adaptive QE	**0.58**	0.23	**0.61**	0.22	**0.47**	0.20

Table 2: QE regression with static and adaptive models

Document	LZA	RTW	WC7
Num. of Sents	2067	2003	2405
	P/R/F-score	P/R/F-score	P/R/F-score
Static QE	0.73/0.08/0.14	0.69/ 0.11/ 0.19	0.74/ 0.10/ 0.18
Adaptive QE	0.69/0.14/**0.24**	0.84/ 0.16/ **0.26**	0.80/ 0.23/ **0.35**

Table 3: Performance on predicting *Good* sentences with static and adaptive models

similarity score of 70% is used as the threshold), therefore only the MT output is returned.

EM sentences are excluded from the study because in general they do not require editing. We focus on the FM and NP sentences[3]. In Table 5 we present the precision, recall and F-score of the "Good" sentences in the FM and NP categories, similar to those shown in Table 3. We consistently observe higher performance on the FM sentences, in terms of precision, recall and F-score. This is expected because these sentences are well covered in the training data. The overall F-score is in line with the test set results shown in Table 3.

We are also interested to know whether the proposed adaptive QE method is helpful to human translators in the MT post-editing task. Based on the TERs predicted by the adaptive QE model, we assign each MT proposal with a confidence label: High ($0 \leq$ TER ≤ 0.2), Medium ($0.2 <$ TER ≤ 0.3), or Low (TER > 0.3). We present the MT proposals with confidence labels to human translators, then measure the percentage of sentences whose MT proposals are used. From Table 6 and 7, we can see that sentences with High and Medium confidence labels are more frequently used by the translators than those with Low labels, for both the FM and NP categories. The MT usage for the FM category is less than that for the NP category because translators can choose FM proposals instead of the MT proposals for correction.

We also measure the translator's productivity gain for MT proposals with different confidence labels. The productivity of a translator is defined as the number of source words translated per unit time. The post editing tool, IBM TranslationManager, records the time that a translator spends on a segment and computes the number of characters that a translator types on the segment so that we can compute how many words the translator has finished in a given time.

[3]The word count distribution of EM, FM and NP is 21%, 38% and 41%, respectively.

Document	LZA	RTW	WC7
TER-Baseline	30.81	30.74	29.96
TER-with Adaptive QE	30.69	30.78	29.56

Table 4: MT Quality with and without Adaptive QE measured by TER

We choose the overall productivity of NP0 as the base unit 1, where there is no proposal presents and the translator has to translate the segments from scratch. Measured with this unit, for example, the overall productivity of FM0 being 1.14 implies a relative gain of 14% over that of NP0, which demonstrates the effectiveness of FM proposals.

Table 6 and 7 also show the productivity gain on sentences with High, Medium and Low labels from FM and NP categories. Again, the productivity gain is consistent with the confidence labels from the adaptive QE model's prediction. The overall productivity gain with confidence-labeled MT proposals is about 10% (comparing FM1 vs. FM0 and NP1 vs. NP0). These results clearly demonstrate the effectiveness of the adaptive QE model in aiding the translators to make use of MT proposals and improve productivity.

Category	Class	FM usage	**MT usage**	**Productivity**
FM1	High	33%	**34%**	**1.35**
	Medium	47%	**18%**	**1.21**
	Low	60%	**8%**	**1.20**
	Overall	45%	**21%**	**1.26**
FM0	High	53%	-	1.12
	Medium	64%	-	1.14
	Low	67%	-	1.16
	Overall	59%	-	1.14

Table 6: MT proposal usage and productivity gain in FM category.
In FM1, both Fuzzy Match and MT proposals present. In control class FM0, only Fuzzy Match proposals present, and therefore, MT usage is not available for FM0. Strong correlation is observed between predicted "High" , "Medium" and "Low" sentences with MT usage and post editing productivity.

Category	Class	**MT usage**	**Productivity**
NP1	High	**50%**	**1.25**
	Medium	**42%**	**1.08**
	Low	**27%**	**1.00**
	Overall	**38%**	**1.09**
NP0	High	-	1.08
	Medium	-	1.00
	Low	-	0.96
	Overall	-	1.00

Table 7: MT proposal usage and productivity gain in NP category.
In NP1, MT is the only proposal available, while in control NP0, there presents no proposal at all and the translator has to translate from scratch. Strong correlation is observed between predicted "High" , "Medium" and "Low" sentences with MT usage and post editing productivity

Type	Precision	Recall	F-score
FM	0.71	0.23	0.35
NP	0.67	0.18	0.29
Overall	**0.69**	**0.21**	**0.32**

Table 5: Performance on predicting *Good* sentences (TER ≤ 0.1) by adaptive QE model

7 Discussion and Conclusion

In this paper we proposed a method to adaptively train a quality estimation model for document-specific MT post editing. With the 26 proposed features derived from decoding process and source sentence syntactic analysis, the proposed QE model achieved better TER prediction, higher correlation with human correction of MT output and higher F-score in finding good translations. The proposed adaptive QE model is deployed to a large scale English-to-Japanese MT post editing project, showing strong correlation with human preference and leading to about 10% gain in human translator productivity.

The training data for QE model can be selected independent of the input document. With such fixed QE training data, it is possible to measure the consistency of the trained QE models, and to allow the sanity check of the document-specific MT models. However, adding such data in the subsampling process extracts more bilingual data for building the MT models, which slightly increase the model building time but increased the translation quality. Another option is to select the sentence pairs from the MT system subsampled training data, which is more similar to the input document thus the trained QE model could be a better match to the input document. However, the QE model training data is no longer constant. The model consistency is no longer guaranteed, and the QE training data must be removed from the MT system training data to avoid data contamination.

References

Nguyen Bach, Fei Huang, and Yaser Al-Onaizan. 2011. Goodness: A method for measuring machine translation confidence. In *ACL*, pages 211–219.

Ergun Biçici, Declan Groves, and Josef van Genabith. 2013. Predicting sentence translation quality using extrinsic and language independent features. *Machine Translation*.

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. 2004. Confidence estimation for machine translation. In *Proceedings of the 20th international conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ondrej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Eighth Workshop on Statistical Machine Translation*, WMT-2013, pages 1–44, Sofia, Bulgaria.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 workshop on statistical machine translation. In *Seventh Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada.

Mariano Felice and Lucia Specia. 2012. Linguistic features for quality estimation. In *Seventh Workshop on Statistical Machine Translation*, pages 96–103, Montréal, Canada.

George Foster, Cyril Goutte, and Roland Kuhn. 2010. Discriminative instance weighting for domain adaptation in statistical machine translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 451–459, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jesús González-Rubio, Jose Ramón Navarro-Cerdán, and Francisco Casacuberta. 2013. Dimensionality reduction methods for machine translation quality estimation. *Machine Translation*, 27(3-4):281–301.

Spence Green, Jeffrey Heer, and Christopher D. Manning. 2013. The efficacy of human post-editing for language translation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 439–448, New York, NY, USA. ACM.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Abraham Ittycheriah and Salim Roukos. 2005. A maximum entropy word aligner for arabic-english machine translation. In *In Proceedings of HLT-EMNLP*, pages 89–96.

Abraham Ittycheriah and Salim Roukos. 2007. Direct translation model 2. In *In HLT-NAACL 2007: Main Conference*, pages 57–64.

Yajuan Lu, Jin Huang, and Qun Liu. 2007. Improving statistical machine translation performance by

training data selection and optimization. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 343–350, Prague, Czech Republic, June. Association for Computational Linguistics.

Spyros Matsoukas, Antti-Veikko I. Rosti, and Bing Zhang. 2009. Discriminative corpus weight estimation for machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*, EMNLP '09, pages 708–717, Stroudsburg, PA, USA. Association for Computational Linguistics.

Christopher B. Quirk. 2004. Training a sentence-level machine translation confidence measure. In *In Proceedings of LREC*.

Sylvain Raybaud, Caroline Lavecchia, David Langlois, and Kamel Smaïli. 2009. New confidence measures for statistical machine translation. *CoRR*, abs/0902.1033.

Salim Roukos, Abraham Ittycheriah, and Jian-Ming Xu. 2012. Document-specific statistical machine translation for improving human translation productivity. In *Proceedings of the 13th international conference on Computational Linguistics and Intelligent Text Processing - Volume Part II*, CICLing'12, pages 25–39, Berlin, Heidelberg. Springer-Verlag.

Alberto Sanchis, Alfons Juan, Enrique Vidal, and Departament De Sistemes Informtics. 2007. Estimation of confidence measures for machine translation. In *In Procedings of Machine Translation Summit XI*.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Radu Soricut and Abdessamad Echihabi. 2010a. Trustrank: inducing trust in automatic translations via ranking. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 612–621, Stroudsburg, PA, USA. Association for Computational Linguistics.

Radu Soricut and Abdessamad Echihabi. 2010b. Trustrank: Inducing trust in automatic translations via ranking. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 612–621. Association for Computational Linguistics.

Lucia Specia, Craig Saunders, Marco Turchi, Zhuoran Wang, and John Shawe-taylor. 2009a. Improving the confidence of machine translation quality estimates. In *In Proceedings of MT Summit XII*.

Lucia Specia, Marco Turchi, Zhuoran Wang, John Shawe-Taylor, and Craig Saunders. 2009b. Improving the confidence of machine translation quality estimates.

Nicola Ueffing and Hermann Ney. 2007. Word-level confidence estimation for machine translation. *Computational Linguistics*, 33(1):9–40.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. Hmm-based word alignment in statistical translation. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 2*, COLING '96, pages 836–841, Stroudsburg, PA, USA. Association for Computational Linguistics.

Deyi Xiong, Min Zhang, and Haizhou Li. 2010. Error detection for statistical machine translation using linguistic features. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 604–611, Stroudsburg, PA, USA. Association for Computational Linguistics.

Translation Assistance by Translation of L1 Fragments in an L2 Context

Maarten van Gompel & Antal van den Bosch
Centre for Language Studies
Radboud University Nijmegen
proycon@anaproy.nl

Abstract

In this paper we present new research in translation assistance. We describe a system capable of translating native language (L1) fragments to foreign language (L2) fragments in an L2 context. Practical applications of this research can be framed in the context of second language learning. The type of translation assistance system under investigation here encourages language learners to write in their target language while allowing them to fall back to their native language in case the correct word or expression is not known. These code switches are subsequently translated to L2 given the L2 context. We study the feasibility of exploiting cross-lingual context to obtain high-quality translation suggestions that improve over statistical language modelling and word-sense disambiguation baselines. A classification-based approach is presented that is indeed found to improve significantly over these baselines by making use of a contextual window spanning a small number of neighbouring words.

1 Introduction

Whereas machine translation generally concerns the translation of whole sentences or texts from one language to the other, this study focusses on the translation of native language (henceforth L1) words and phrases, i.e. smaller fragments, in a foreign language (L2) context. Despite the major efforts and improvements, automatic translation does not yet rival human-level quality. Vexing issues are morphology, word-order change and long-distance dependencies. Although there is a morpho-syntactic component in this research, our scope is more constrained; its focus is on the faithful preservation of meaning from L1 to L2, akin to the role of the translation model in Statistical Machine Translation (SMT).

The cross-lingual context in our research question may at first seem artificial, but its design explicitly aims at applications related to computer-aided language learning (Laghos and Panayiotis, 2005; Levy, 1997) and computer-aided translation (Barrachina et al., 2009). Currently, language learners need to refer to a bilingual dictionary when in doubt about a translation of a word or phrase. Yet, this problem arises in a context, not in isolation; the learner may have already translated successfully a part of the text into L2 leading up to the problematic word or phrase. Dictionaries are not the best source to look up context; they may contain example usages, but remain biased towards single words or short expressions.

The proposed application allows code switching and produces context-sensitive suggestions as writing progresses. In this research we test the feasibility of the foundation of this idea.The following examples serve to illustrate the idea and demonstrate what output the proposed translation assistance system would ideally produce. The parts in bold correspond to respectively the inserted fragment and the system translation.

- Input (L1=English,L2=Spanish): *"Hoy vamos a **the swimming pool**."*
  Desired output: *"Hoy vamos a **la piscina**."*
- Input (L1-English, L2=German): *"Das wetter ist wirklich **abominable**."*
  Desired output: *"Das wetter ist wirklich **ekelhaft**."*
- Input (L1=French,L2=English): *"I **rentre à la maison** because I am tired."*
  Desired output: *"I **return home** because I am tired."*
- Input (L1=Dutch, L2=English): *"Workers are facing a massive **aanval op** their employ-*

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 871–880,
Baltimore, Maryland, USA, June 23-25 2014.

ment and social rights."
Desired output: *"Workers are facing a massive **attack on** their employment and social rights."*

The main research question in this research is how to disambiguate an L1 word or phrase to its L2 translation based on an L2 context, and whether such cross-lingual contextual approaches provide added value compared to baseline models that are not context informed or compared to standard language models.

2 Data preparation

Preparing the data to build training and test data for our intended translation assistance system is not trivial, as the type of interactive translation assistant we aim to develop does not exist yet. We need to generate training and test data that realistically emulates the task. We start with a parallel corpus that is tokenised for both L1 and L2. No further linguistic processing such as part-of-speech tagging or lemmatisation takes place in our experiments; adding this remains open for future research.

The parallel corpus is randomly sampled into two large and equally-sized parts. One is the basis for the training set, and the other is the basis for the test set. The reason for such a large test split shall become apparent soon.

From each of the splits (S), a phrase-translation table is constructed automatically in an unsupervised fashion. This is done using the scripts provided by the Statistical Machine Translation system Moses (Koehn et al., 2007). It invokes GIZA++ (Och and Ney, 2000) to establish statistical word alignments based on the IBM Models and subsequently extracts phrases using the `grow-diag-final` algorithm (Och and Ney, 2003). The result, independent for each set, will be a phrase-translation table (T) that maps phrases in L1 to L2. For each phrase-pair (f_s, f_t) this phrase-translation table holds the computed translation probabilities $P(f_s|f_t)$ and $P(f_t|f_s)$.

Given these phrase-translation tables, we can now extract both training data and test data using the algorithm in Figure 1. In our discourse, the source language (s) corresponds to L1, the fallback language used for by the end-user for inserting fragments, whilst the target language (t) is L2.

Step 4 is effectively a filter: two thresholds can be configured to discard weak alignments,

1. using phrase-translation table T and parallel corpus split S
2. **for** each aligned sentence pair $(sentence_s \in S_s, sentence_t \in S_t)$ in the parallel corpus split (S_s,S_t):
3. **for** each fragment $(f_s \in sentence_s, f_t \in sentence_t)$ where $(f_s, f_t) \in T$:
4. **if** $P(f_s|f_t) \cdot P(f_t|f_s) \geq \lambda_1$ **and** $P(f_s|f_t) \cdot P(f_t|f_s) \geq \lambda_2 \cdot P(f_s|f_{strongest\_t}) \cdot P(f_{strongest\_t}|f_s)$:
5. Output a pair $(sentence'_t, sentence_t)$ where $sentence'_t$ is a copy of t but with fragment f_t substituted by f_s, i.e. the introduction of an L1 word or phrase in an L2 sentence.

Figure 1: Algorithm for extracting training and test data on the basis of a phrase-translation table (T) and subset/split from a parallel corpus (S). The indentation indicates the nesting.

i.e. those with low probabilities, from the phrase-translation table so that only strong couplings make it into the generated set. The parameter λ_1 adds a constraint based on the product of the two conditional probabilities ($P(f_t|f_s) \cdot P(f_s|f_t)$), and sets a threshold that has to be surpassed. A second parameter λ_2 further limits the considered phrase pairs (f_s, f_t) to have the product of their conditional probabilities not not deviate more than a fraction λ_2 from the joint probability for the strongest possible pairing for f_s, the source fragment. $f_{strongest\_t}$ in Figure 1 corresponds to the best scoring translation for a given source fragment f_s. This metric thus effectively prunes weaker alternative translations in the phrase-translation table from being considered if there is a much stronger candidate. Nevertheless, it has to be noted that even with λ_1 and λ_2, the test set will include a certain amount of errors. This is due to the nature of the unsupervised method with which the phrase-translation table is constructed. For our purposes however, the test set suffices to test our hypothesis.

In our experiments, we choose fixed values for these parameters, by manual inspection and judgement of the output. The λ_1 parameter was set to 0.01 and λ_2 to 0.8. Whilst other thresholds may possibly produce cleaner sets, this is hard to evaluate as finding optimal values causes a prohibitive increase in complexity of the search space, and again this is not necessary to test our hypothesis.

The output of the algorithm in Figure 1 is a modified set of sentence pairs $(sentence'_t, sentence_t)$, in which the same sentence pair may be used multiple times with different L1 substitutions for different fragments. The final test set is created by randomly sampling the desired number of test instances.

Note that the training set and test set are constructed on their own respective and independently generated phrase-translation tables. This ensures complete independence of training and test data. Generating test data using the same phrase-translation table as the training data would introduce a bias. The fact that a phrase-translation table needs to be constructed for the test data is also the reason that the parallel corpus split from which the test data is derived has to be large enough, ensuring better quality.

We concede that our current way of testing is a mere approximation of the real-world scenario. An ideal test corpus would consist of L2 sentences with L1 fallback as crafted by L2 language learners with an L1 background. However, such corpora do not exist as yet. Nevertheless, we hope to show that our automated way of test set generation is sufficient to test the feasibility of our core hypothesis that L1 fragments can be translated to L2 using L2 context information.

3 System

We develop a classifier-based system composed of so-called "classifier experts". Numerous classifiers are trained and each is an expert in translating a single word or phrase. In other words, for each word type or phrase type that occurs as a fragment in the training set, and which does not map to just a single translation, a classifier is trained. The classifier maps the L1 word or phrase in its L2 context to its L2 translation. Words or phrases that always map to a single translation are stored in a simple mapping table, as a classifier would have no added value in such cases. The classifiers use the IB1 algorithm (Aha et al., 1991) as implemented in TiMBL (Daelemans et al., 2009).[1] IB1 implements k-nearest neighbour classification. The choice for this algorithm is motivated by the fact that it handles multiple classes with ease, but first and foremost because it has been successfully employed for word sense disambiguation in other studies (Hoste et al., 2002; Decadt et al., 2004), in particular in cross-lingual word sense disambiguation, a task closely resembling our current task (van Gompel and van den Bosch, 2013). It has also been used in machine translation studies in which local source context is used to classify source phrases into target phrases, rather than looking them up in a phrase table (Stroppa et al., 2007; Haque et al., 2011). The idea of local phrase selection with a discriminative machine learning classifier using additional local (source-language) context was introduced in parallel to Stroppa *et al.* (2007) by Carpuat and Wu (2007) and Giménez and Márquez (2007); cf. Haque *et al.* (2011) for an overview of more recent methods.

The feature vector for the classifiers represents a local context of neighbouring words, and optionally also global context keywords in a binary-valued bag-of-words configuration. The local context consists of an X number of L2 words to the left of the L1 fragment, and Y words to the right.

When presented with test data, in which the L1 fragment is explicitly marked, we first check whether there is ambiguity for this L1 fragment and if a direct translation is available in our simple mapping table. If so, we are done quickly and need not rely on context information. If not, we check for the presence of a classifier expert for the offered L1 fragment; only then we can proceed by extracting the desired number of L2 local context words to the immediate left and right of this fragment and adding those to the feature vector. The classifier will return a probability distribution of the most likely translations given the context and we can replace the L1 fragment with the highest scoring L2 translation and present it back to the user.

In addition to local context features, we also experimented with global context features. These are a set of L2 contextual keywords for each L1 word/phrase and its L2 translation occurring in the same sentence, not necessarily in the immediate neighbourhood of the L1 word/phrase. The keywords are selected to be indicative for a specific

[1] `http://ilk.uvt.nl/timbl`

translation. We used the method of extraction by Ng and Lee (1996) and encoded all keywords in a binary bag of words model. The experiments however showed that inclusion of such keywords did not make any noticeable impact on any of the results, so we restrict ourselves to mentioning this negative result.

Our full system, including the scripts for data preparation, training, and evaluation, is implemented in Python and freely available as open-source from `http://github.com/proycon/colibrita/` . Version tag `v0.2.1` is representative for the version used in this research.

3.1 Language Model

We also implement a statistical language model as an optional component of our classifier-based system and also as a baseline to compare our system to. The language model is a trigram-based backoff language model with Kneser-Ney smoothing, computed using SRILM (Stolcke, 2002) and trained on the same training data as the translation model. No additional external data was brought in, to keep the comparison fair.

For any given hypothesis H, results from the L1 to L2 classifier are combined with results from the L2 language model. We do so by normalising the class probability from the classifier ($score_T(H)$), which is our translation model, and the language model ($score_{lm}(H)$), in such a way that the highest classifier score for the alternatives under consideration is always 1.0, and the highest language model score of the sentence is always 1.0. Take $score_T(H)$ and $score_{lm}(H)$ to be log probabilities, the search for the best (most probable) translation hypothesis $\hat{H}$ can then be expressed as:

$$\hat{H} = \arg\max_H (score_T(H) + score_{lm}(H)) \quad (1)$$

If desired, the search can be parametrised with variables λ_3 and λ_4, representing the weights we want to attach to the classifier-based translation model and the language model, respectively. In the current study we simply left both weights set to one, thereby assigning equal importance to translation model and language model.

4 Evaluation

Several automated metrics exist for the evaluation of L2 system output against the L2 reference output in the test set. We first measure absolute accuracy by simply counting all output fragments that exactly match the reference fragments, as a fraction of the total amount of fragments. This measure may be too strict, so we add a more flexible *word accuracy* measure which takes into account partial matches at the word level. If output o is a subset of reference r then a score of $\frac{|o|}{|r|}$ is assigned for that sentence pair. If instead, r is a subset of o, then a score of $\frac{|r|}{|o|}$ will be assigned. A perfect match will result in a score of 1 whereas a complete lack of overlap will be scored 0. The word accuracy for the entire set is then computed by taking the sum of the word accuracies per sentence pair, divided by the total number of sentence pairs.

We also compute a recall metric that measures the number of fragments that the system provided a translation for as a fraction of the total number of fragments in the input, regardless of whether the fragment is translated correctly or not. The system may skip fragments for which it can find no solution at all.

In addition to these, the system's output can be compared against the L2 reference translation(s) using established Machine Translation evaluation metrics. We report on BLEU, NIST, METEOR, and word error rate metrics WER and PER. These scores should generally be much better than the typical MT system performances as only local changes are made to otherwise "perfect" L2 sentences.

5 Baselines

A context-insensitive yet informed baseline was constructed to assess the impact of L2 context information in translating L1 fragments. The baseline selects the most probable L1 fragment per L2 fragment according to the phrase-translation table. This baseline, henceforth referred to as the 'most likely fragment' baseline (MLF) is analogous to the 'most frequent sense'-baseline common in evaluating WSD systems.

A second baseline was constructed by weighing the probabilities from the translation table directly with the L2 language model described earlier. It adds a LM component to the MLF baseline. This LM baseline allows the comparison of classification through L1 fragments in an L2 context, with a more traditional L2 context modelling (i.e. target language modelling) which is also cus-

tomary in MT decoders. Computing this baseline is done in the same fashion as previously illustrated in Equation 1, where $score_T$ then represents the normalised $p(t|s)$ score from the phrase-translation table rather than the class probability from the classifier.

6 Experiments & Results

The data for our experiments were drawn from the Europarl parallel corpus (Koehn, 2005) from which we extracted two sets of $200,000$ sentence pairs each for several language pairs. These were used to form the training and test sets. The final test sets are a randomly sampled $5,000$ sentence pairs from the $200,000$-sentence test split for each language pair.

All input data for the experiments in this section are publicly available[2].

Let us first zoom in to convey a sense of scale on a specific language pair. The actual Europarl training set we generate for English (L1) to Spanish (L2), i.e. English fallback in a Spanish context, consists of $5,608,015$ sentence pairs. This number is much larger than the $200,000$ we mentioned before because single sentence pairs may be reused multiple times with different marked fragments. From this training set of sentence pairs over $100,000$ classifier experts are derived. The eleven largest classifiers are shown in Table 1, along with the number of training instances per classifier. The full table would reveal a Zipfian distribution.

Fragment	Training instances	Translations
the	256,772	la, el, los, las
of	139,273	de, del
and	128,074	y, de, e
to	66,565	a, para, que, de
a	54,306	un, una
is	40,511	es, está, se
for	34,054	para, de, por
this	29,691	este, esta, esto
European	26,543	Europea, Europeo Europeas, Europeos
on	23,147	sobre, en
of the	22,361	de la, de los

Table 1: The top eleven classifier experts for English to Spanish. The eleventh entry is included as an example of a common phrasal fragment

Among the classifier experts are only words and phrases that are ambiguous and may thus map to multiple translations. This implies that such words and phrases must have occurred at least twice in the corpus, though this threshold is made configurable and could have been set higher to limit the number of classifiers. The remaining $246,380$ unambiguous mappings are stored in a separate mapping table.

[2]Download and unpack `http://lst.science.ru.nl/~proycon/colibrita-acl2014-data.zip`

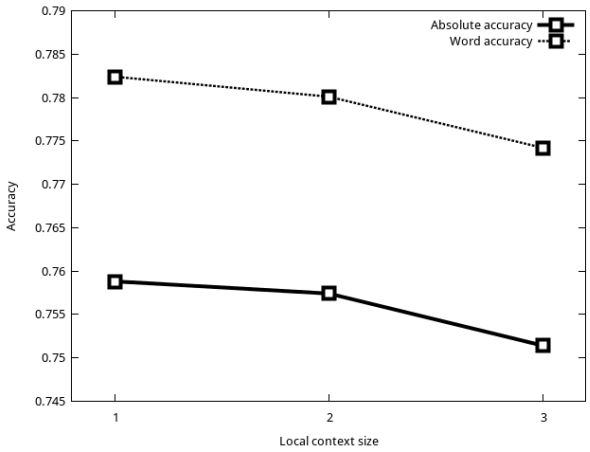


Figure 2: Accuracy for different local context sizes, Europarl English to Spanish

For the classifier-based system, we tested various different feature vector configurations. The first experiment, of which the results are shown in Figure 2, sets a fixed and symmetric local context size across all classifiers, and tests three context widths. Here we observe that a context width of one yields the best results. The BLEU scores, not included in the figure but shown in Table 2, show a similar trend. This trend holds for all the MT metrics.

Table 2 shows the results for English to Spanish in more detail and adds a comparison with the two baseline systems. The various `lXrY` configurations use the same feature vector setup for all classifier experts. Here X indicates the left context size and Y the right context size. The `auto` configuration does not uniformly apply the same feature vector setup to all classifier experts but instead seeks to find the optimal setup per classifier expert. This shall be further discussed in Section 6.1.

As expected, the LM baseline substantially outperforms the context-insensitive MLF baseline. Second, our classifier approach attains a substantially higher accuracy than the LM baseline. Third, we observe that adding the language model to our classifier leads to another significant gain

Configuration	Accuracy	Word Accuracy	BLEU	METEOR	NIST	WER	PER
MLF baseline	0.6164	0.6662	0.972	0.9705	17.0784	1.4465	1.4209
LM baseline	0.7158	0.7434	0.9785	0.9739	17.1573	1.1735	1.1574
l1r1	0.7588	0.7824	0.9801	0.9747	17.1550	1.1625	1.1444
l2r2	0.7574	0.7801	0.9800	0.9746	17.1550	1.1750	1.1569
l3r3	0.7514	0.7742	0.9796	0.9744	17.1445	1.1946	1.1780
l1r1+LM	**0.7810**	**0.7973**	**0.9816**	**0.9754**	**17.1685**	**1.0946**	**1.077**
auto	0.7626	0.7850	0.9803	0.9748	17.1544	1.1594	1.1424
auto+LM	0.7796	0.7966	0.9815	0.9754	17.1664	1.1021	1.0845
l1r0	0.6924	0.7223	0.9757	0.9723	17.1087	1.3415	1.3249
l2r0	0.6960	0.7245	0.9759	0.9724	17.1091	1.3364	1.3193
l2r1	0.7624	0.7849	0.9803	0.9748	17.1558	1.1554	1.1378

Table 2: Europarl results for English to Spanish (i.e English fallback in Spanish context). Recall = 0.9422

(configuration `l1r1+LM` in the results in Table 2). It appears that the classifier approach and the L2 language model are able to complement each other.

Statistical significance on the BLEU scores was tested using pairwise bootstrap sampling (Koehn, 2004). All significance tests were performed with $5,000$ iterations. We compared the outcomes of several key configurations. We first tested `l1r1` against both baselines; both differences are significant at $p < 0.01$ for both. The same significance level was found when comparing `l1r1+LM` against `l1r1`, `auto+LM` against `auto`, as well as the LM baseline against the MLF baseline. Automatic feature selection `auto` was found to perform statistically better than `l1r1`, but only at $p < 0.05$. Conclusions with regard to context width may have to be tempered somewhat, as the performance of the `l1r1` configuration was found to not be significantly better than that of the `l2r2` configuration. However, `l1r1` performs significantly better than `l3r3` at $p < 0.01$, and `l2r2` performs significantly better than `l3r3` at $p < 0.01$.

In Table 3 we present some illustrative examples from the English→Spanish Europarl data. We show the difference between the most-likely-fragment baseline and our system.

Likewise, Table 4 exemplifies small fragments from the `l1r1` configuration compared to the same configuration enriched with a language model. We observe in this data that the language model often has the added power to choose a correct translation that is not the first prediction of the classifier, but one of the weaker alternatives that nevertheless fits better. Though the classifier generally works best in the `l1r1` configuration, i.e. with context size one, the trigram-based language model allows further left-context information to be incorporated that influences the weights of the classifier output, successfully forcing the system to select alternatives. This combination of a classifier with context size one and trigram-based language model proves to be most effective and reaches the best results so far. We have not conducted experiments with language models of other orders.

6.1 Context optimisation

It has been argued that classifier experts in a word sense disambiguation ensemble should be individually optimised (Decadt et al., 2004; van Gompel and van den Bosch, 2013). The latter study on cross-lingual WSD finds a positive impact when conducting feature selection per classifier. This intuitively makes sense; a context of one may seem to be better than any other when uniformly applied to all classifier experts, but it may well be that certain classifiers benefit from different feature selections. We therefore proceed with this line of investigation as well.

Automatic configuration selection was done by performing leave-one-out testing (for small number of instances) or 10-fold-cross validation (for larger number of instances, $n \geq 20$) on the training data per classifier expert. Various configurations were tested. Per classifier expert, the best scoring configuration was selected, referred to as the `auto` configuration in Table 2. The `auto` configuration improves results over the uniformly

Input: Mientras no haya prueba en contrario , la financiación de partidos políticos **European** sólo se justifica , incluso después del tratado de Niza , desde el momento en que concurra a la expresión del sufragio universal , que es la única definición aceptable de un partido político .
MLF baseline: Mientras no haya prueba en contrario , la financiación de partidos políticos **Europea** sólo se justifica , incluso después del tratado de Niza , desde el momento en que concurra a la expresión del sufragio universal , que es la única definición aceptable de un partido político .
l1r1: Mientras no haya prueba en contrario , la financiación de partidos políticos **europeos** sólo se justifica , incluso después del tratado de Niza , desde el momento en que concurra a la expresión del sufragio universal , que es la única definición aceptable de un partido político .

Input: Esta Directiva es nuestra oportunidad **to** marcar una verdadera diferencia , reduciendo la trágica pérdida de vidas en nuestras carreteras .
MLF baseline: Esta Directiva es nuestra oportunidad **a** marcar una verdadera diferencia , reduciendo la trágica pérdida de vidas en nuestras carreteras .
l1r1: Esta Directiva es nuestra oportunidad **para** marcar una verdadera diferencia , reduciendo la trágica pérdida de vidas en nuestras carreteras .

Input: Es la **last** vez que me dirijo a esta Cámara .
MLF baseline: Es la **pasado** vez que me dirijo a esta Cámara .
l1r1: Es la **última** vez que me dirijo a esta Cámara .

Input: Pero el enfoque actual de la Comisión no puede conducir a una buena política ya que es tributario del funcionamiento del mercado y de las normas establecidas por la OMC , el FMI y el Banco Mundial , normas que siguen siendo desfavorables para los **developing countries** .
MLF baseline: Pero el enfoque actual de la Comisión no puede conducir a una buena política ya que es tributario del funcionamiento del mercado y de las normas establecidas por la OMC , el FMI y el Banco Mundial , normas que siguen siendo desfavorables para los **los países en desarrollo** .
l1r1: Pero el enfoque actual de la Comisión no puede conducir a una buena política ya que es tributario del funcionamiento del mercado y de las normas establecidas por la OMC , el FMI y el Banco Mundial , normas que siguen siendo desfavorables para los **países en desarrollo** .

Table 3: Some illustrative examples of MLF-baseline output versus system output, in which system output matches the correct human reference output. The actual fragments concerned are highlighted in bold. The first example shows our system correcting for number agreement, the second a correction in selecting the right preposition, and the third shows that the English word *last* can be translated in different ways, only one of which is correct in this context. The last example shows a phrasal translation, in which the determiner was duplicated in the baseline

applied feature selection. However, if we enable the language model as we do in the `auto+LM` configuration we do not notice an improvement over `l1r1+LM`, surprisingly. We suspect the lack of impact here can be explained by the trigram-based Language Model having less added value when the (left) context size of the classifier is two or three; they are now less complementary.

Table 5 lists what context sizes have been chosen in the automatic feature selection. A context size of one prevails in the vast majority of cases, which is not surprising considering the good results we have already seen with this configuration.

In this study we did not yet conduct optimisation of the classifier parameters. We used the IB1 algorithm with $k = 1$ and the default values of the TiMBL implementation. In earlier work van Gompel and van den Bosch (2013), we reported a decrease in performance due to overfitting when this is done, so we do not expect it to make a positive impact. The second reason for omitting this is more practical in nature; to do this in combination with feature selection would add substantial search complexity, making experiments far more time consuming, even prohibitively so.

66.5%	l1r1
19.9%	l2r2
7.7%	l3r3
3.5%	l4r4
2.4%	l5r5

Table 5: Frequency of automatically selected configurations on English to Spanish Europarl dataset

The bottom lines in Table 2 represent results when all right-context is omitted, emulating a real-time prediction when no right context is available yet. This has a substantial negative impact on re-

Input: Sin ese tipo de protección la gente no aprovechará la oportunidad **to** vivir , viajar y trabajar donde les parezca en la Unión Europea .
l1r1: Sin ese tipo de protección la gente no aprovechará la oportunidad **para** vivir , viajar y trabajar donde les parezca en la Unión Europea .
l1r1+LM: Sin ese tipo de protección la gente no aprovechará la oportunidad **de** vivir , viajar y trabajar donde les parezca en la Unión Europea .

Input: La Comisión también está acometiendo medidas en el ámbito social y **educational** con vistas a mejorar la situación de los niños .
l1r1: La Comisión también está acometiendo medidas en el ámbito social y **educativas** con vistas a mejorar la situación de los niños .
l1r1+LM: La Comisión también está acometiendo medidas en el ámbito social y **educativo** con vistas a mejorar la situación de los niños .

Table 4: Some examples of l1r1 versus the same configuration enriched with a language model.

sults. We experimented with several asymmetric configurations and found that taking two words to the left and one to the right yields even better results than symmetric configurations for this data set. This result is in line with the positive effect of adding the LM to the `l1r1`.

In order to draw accurate conclusions, experiments on a single data set and language pair are not sufficient. We therefore conducted a number of experiments with other language pairs, and present the abridged results in Table 6.

There are some noticeable discrepancies for some experiments in Table 6 when compared to our earlier results in Table 2. We see that the language model baseline for English→French shows the same substantial improvement over the baseline as our English→Spanish results. The same holds for the Chinese→English experiment. However, for English→Dutch and English→Chinese we find that the LM baseline actually performs slightly worse than baseline. Nevertheless, in all these cases, the positive effect of including a Language Model to our classifier-based system again shows. Also, we note that in all cases our system performs better than the two baselines.

Another discrepancy is found in the BLEU scores of the English→Chinese experiments, where we measure an unexpected drop in BLEU score under baseline. However, all other scores do show the expected improvement. The error rate metrics show improvement as well. We therefore attach low importance to this deviation in BLEU here.

In all of the aforementioned experiments, the system produced a single solution for each of the fragments, the one it deemed best, or no solution at all if it could not find any. Alternative evaluation metrics could allow the system to output multiple alternatives. Omission of a solution by definition causes a decrease in recall. In all of our experiments recall is high (well above 90%), mostly because train and test data lie in the same domain and have been generated in the same fashion, lower recall is expected with more real-world data.

7 Discussion and conclusion

In this study we have shown the feasibility of a classifier-based translation assistance system in which L1 fragments are translated in an L2 context, in which the classifier experts are built individually per word or phrase. We have shown that such a translation assistance system scores both above a context-insensitive baseline, as well as an L2 language model baseline.

Furthermore, we found that combining this cross-language context-sensitive technique with an L2 language model boosts results further.

The presence of a one-word right-hand side context proves crucial for good results, which has implications for practical translation assistance application that translate as soon as the user finishes an L1 fragment. Revisiting the translation when right context becomes available would be advisable.

We tested various configurations and conclude that small context sizes work better than larger ones. Automated configuration selection had positive results, yet the system with context size one and an L2 language model component often produces the best results. In static configurations, the failure of a wider context window to be more suc-

Dataset	L1	L2	Configuration	Accuracy	Word Accuracy	BLEU
europarl200k	en	nl	baseline	0.7026	0.7283	0.9771
europarl200k	en	nl	LM baseline	0.6958	0.7195	0.9773
europarl200k	en	nl	l1r1	0.7790	0.7941	0.9814
europarl200k	en	nl	l1r1+LM	**0.7838**	**0.7973**	**0.9818**
europarl200k	en	nl	auto	0.7796	0.7947	0.9815
europarl200k	en	nl	auto+LM	0.7812	0.7954	0.9816
europarl200k	en	fr	baseline	0.5874	0.6403	0.9709
europarl200k	en	fr	LM baseline	0.7054	0.7319	0.9787
europarl200k	en	fr	l1r1	0.7416	0.7698	0.9797
europarl200k	en	fr	l1r1+LM	**0.7680**	**0.7885**	**0.9815**
europarl200k	en	fr	auto	0.7484	0.7737	0.9801
europarl200k	en	fr	auto+LM	0.7654	0.7860	0.9813
iwslt12ted	en	zh	baseline	0.6622	0.7122	**0.6421**
iwslt12ted	en	zh	LM baseline	0.6550	0.6982	0.6416
iwslt12ted	en	zh	l1r1	0.7150	0.7531	0.5736
iwslt12ted	en	zh	l1r1+LM	**0.7296**	**0.7619**	0.5826
iwslt12ted	en	zh	auto	0.7150	0.7519	0.5746
iwslt12ted	en	zh	auto+LM	0.7280	0.7605	0.5833
iwslt12ted	zh	en	baseline	0.5784	0.6167	0.9634
iwslt12ted	zh	en	LM baseline	0.6148	0.6463	0.9656
iwslt12ted	zh	en	l1r1	0.7104	0.7338	0.9709
iwslt12ted	zh	en	l1r1+LM	**0.7270**	**0.7460**	**0.9721**
iwslt12ted	zh	en	auto	0.7078	0.7319	0.9709
iwslt12ted	zh	en	auto+LM	0.7230	0.7428	0.9719

Table 6: Results on different datasets and language pairs. The `iwslt12ted` set is the dataset used in the IWSLT 2012 Evaluation Campaign (Federico et al., 2012), and is formed by a collection of transcriptions of TED talks. Here we used of just over $70,000$ sentences for training. Recall for each of the four datasets is 0.9498 (en-nl), 0.9494 (en-fr), 0.9386 (en-zh), and 0.9366 (zh-en)

cesful may be attributed to the increased sparsity that comes from such an expansion.

The idea of a comprehensive translation assistance system may extend beyond the translation of L1 fragments in an L2 context. There are more NLP components that might play a role if such a system were to find practical application. Word completion or predictive editing (in combination with error correction) would for instance seem an indispensable part of such a system, and can be implemented alongside the technique proposed in this study. A point of more practically-oriented future research is to see how feasible such combinations are and what techniques can be used.

An application of our idea outside the area of translation assistance is post-correction of the output of some MT systems that, as a last-resort heuristic, copy source words or phrases into their output, producing precisely the kind of input our system is trained on. Our classification-based approach may be able to resolve some of these cases operating as an add-on to a regular MT system – or as a independent post-correction system.

Our system allows L1 fragments to be of arbitrary length. If a fragment was not seen during training stage, and is therefore not covered by a classifier expert, then the system will be unable to translate it. Nevertheless, if a longer L1 fragment can be decomposed into subfragments that are known, then some recombination of the translations of said sub-fragments may be a good translation for the whole. We are currently exploring this line of investigation, in which the gap with MT narrows further.

Finally, an important line of future research is the creation of a more representative test set. Lacking an interactive system that actually does what we emulate, we hypothesise that good approximations would be to use gap exercises, or cloze tests, that test specific aspects difficulties in language learning. Similarly, we may use L2 learner corpora with annotations of code-switching points or errors. Here we then assume that places where L2 errors occur may be indicative of places where L2 learners are in some trouble, and might want to fall back to generating L1. By then manually translating gaps or such problematic fragments into L1 we hope to establish a more realistic test set.

References

D. W. Aha, D. Kibler, and M. K. Albert. 1991. Instance-based learning algorithms. *Machine*

Learning, 06(1):37–66, January.

S. Barrachina, O. Bender, F. Casacuberta, J. Civera, E. Cubel, S. Khadivi, A. L. Lagarda, H. Ney, J. Tomás, E. Vidal, and J.M. Vilar. 2009. Statistical approaches to computer-assisted translation. *Computational Linguistics*, 35(1):3–28.

M. Carpuat and D. Wu. 2007. Improving statistical machine translation using word sense disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 61–72.

W. Daelemans, J. Zavrel, K. van der Sloot, and A. van den Bosch. 2009. TiMBL: Tilburg memory based learner, version 6.2, reference guide. Technical Report ILK 09-01, ILK Research Group, Tilburg University.

B. Decadt, V. Hoste, W. Daelemans, and A. van den Bosch. 2004. GAMBL, genetic algorithm optimization of memory-based WSD. In R. Mihalcea and P. Edmonds, editors, *Proceedings of the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (Senseval-3)*, pages 108–112, New Brunswick, NJ. ACL.

M. Federico, M. Cettolo, L. Bentivogli, M. Paul, and S. Stüker. 2012. Overview of the IWSLT 2012 evaluation campaign. In *Proceedings of the seventh International Workshop on Spoken Language Translation (IWSLT)*, pages 12–33.

J. Giménez and L. Màrquez. 2007. Context-aware discriminative phrase selection for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 159–166, Prague, Czech Republic, June. Association for Computational Linguistics.

R. Haque, S. Kumar Naskar, A. van den Bosch, and A. Way. 2011. Integrating source-language context into phrase-based statistical machine translation. *Machine Translation*, 25(3):239–285, September.

V. Hoste, I. Hendrickx, W. Daelemans, and A. van den Bosch. 2002. Parameter optimization for machine learning of word sense disambiguation. *Natural Language Engineering*, 8(4):311–325.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June. Association for Computational Linguistics.

P. Koehn. 2004. Statistical significance tests for machine translation evaluation. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain, July. Association for Computational Linguistics.

P. Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *In Proceedings of the Machine Translation Summit X ([MT]'05).*, pages 79–86.

A. Laghos and Z. Panayiotis. 2005. Computer assisted/aided language learning. pages 331–336.

M. Levy. 1997. *Computer-assisted language learning: Context and conceptualization.* Oxford: Clarendon Press.

H. Tou Ng and H. Beng Lee. 1996. Integrating multiple knowledge sources to disambiguate word sense: An exemplar-based approach. In *ACL*, pages 40–47.

F.J. Och and H. Ney. 2000. Giza++: Training of statistical translation models. Technical report, RWTH Aachen, University of Technology.

F. J. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Comput. Linguist.*, 29(1):19–51, March.

A. Stolcke. 2002. Srilm - an extensible language modeling toolkit. In John H. L. Hansen and Bryan L. Pellom, editors, *7th International Conference on Spoken Language Processing, ICSLP2002 - INTERSPEECH 2002, Denver, Colorado, USA, September 16-20, 2002*. ISCA.

N. Stroppa, A. van den Bosch, and A. Way. 2007. Exploiting source similarity for SMT using context-informed features. In A. Way and B. Gawronska, editors, *Proceedings of the 11th International Conference on Theoretical Issues in Machine Translation (TMI 2007)*, pages 231–240, Skövde, Sweden.

M. van Gompel and A. van den Bosch. 2013. WSD2: Parameter optimisation for memory-based cross-lingual word-sense disambiguation. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval 2013), in conjunction with the Second Joint Conference on Lexical and Computational Semantics*.

Response-based Learning for Grounded Machine Translation

Stefan Riezler and **Patrick Simianer** and **Carolin Haas**
Department of Computational Linguistics
Heidelberg University, 69120 Heidelberg, Germany
{riezler,simianer,haas1}@cl.uni-heidelberg.de

Abstract

We propose a novel learning approach for statistical machine translation (SMT) that allows to extract supervision signals for structured learning from an extrinsic response to a translation input. We show how to generate responses by grounding SMT in the task of executing a semantic parse of a translated query against a database. Experiments on the GEOQUERY database show an improvement of about 6 points in F1-score for response-based learning over learning from references only on returning the correct answer from a semantic parse of a translated query. In general, our approach alleviates the dependency on human reference translations and solves the reachability problem in structured learning for SMT.

1 Introduction

In this paper, we propose a novel approach for learning and evaluation in statistical machine translation (SMT) that borrows ideas from response-based learning for grounded semantic parsing. In this framework, the meaning of a sentence is defined in the context of an extrinsic task. Successful communication of meaning is measured by a successful interaction in this task, and feedback from this interaction is used for learning.

We suggest that in a similar way the preservation of meaning in machine translation should be defined in the context of an interaction in an extrinsic task. For example, in the context of a game, a description of a game rule is translated successfully if correct game moves can be performed based only on the translation. In the context of a question-answering scenario, a question is translated successfully if the correct answer is returned based only on the translation of the query.

We propose a framework of response-based learning that allows to extract supervision signals for structured learning from the response of an extrinsic task to a translation input. Here, learning proceeds by "trying out" translation hypotheses, receiving a response from interacting in the task, and converting this response into a supervision signal for updating model parameters. In case of positive feedback, the predicted translation can be treated as reference translation for a structured learning update. In case of negative feedback, a structural update can be performed against translations that have been approved previously by positive task feedback. This framework has several advantages:

- The supervision signal in response-based learning has a different quality than supervision by human-generated reference translations. While a human reference translation is generated independently of the SMT task, conversion of predicted translations into references is always done with respect to a specific task. In this sense we speak of grounding meaning transfer in an extrinsic task.

- Response-based learning can repeatedly try out system predictions by interacting in the extrinsic task. Instead of and in addition to learning from human reference translations, response-based learning allows to convert multiple system translations into references. This alleviates the supervision problem in cases where parallel data are scarce.

- Task-specific response acts upon system translations. This avoids the problem of unreachability of independently generated reference translations by the SMT system.

The proposed approach of response-based learning opens the doors for various extrinsic tasks

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 881–891,
Baltimore, Maryland, USA, June 23-25 2014.

in which SMT systems can be trained and evaluated. In this paper, we present a proof-of-concept experiment that uses feedback from a simulated world environment. Building on prior work in grounded semantic parsing, we generate translations of queries, and receive feedback by executing semantic parses of translated queries against the database. Successful response is defined as receiving the same answer from the semantic parses for the translation and the original query. Our experimental results show an improvement of about 6 points in F1-score for response-based learning over standard structured learning from reference translations. We show in an error analysis that this improvement can be attributed to using structural and lexical variants of reference translations as positive examples in response-based learning. Furthermore, translations produced by response-based learning are found to be grammatical. This is due to the possibility to boost similarity to human reference translations by the additional use of a cost function in our approach.

2 Related Work

The key idea of *grounded language learning* is to study natural language in the context of a non-linguistic environment, in which meaning is grounded in perception and/or action. This presents an analogy to human learning, where a learner tests her understanding in an actionable setting. Such a setting can be a simulated world environment in which the linguistic representation can be directly executed by a computer system. For example, in semantic parsing, the learning goal is to produce and successfully execute a meaning representation. Executable system actions include access to databases such as the GEOQUERY database on U.S. geography (Wong and Mooney (2006), *inter alia*), the ATIS travel planning database (Zettlemoyer and Collins (2009), *inter alia*), robotic control in simulated navigation tasks (Chen and Mooney (2011), *inter alia*), databases of simulated card games (Goldwasser and Roth (2013), *inter alia*), or the user-generated contents of FREEBASE (Cai and Yates (2013), *inter alia*). Since there are many possible correct parses, matching against a single gold standard falls short of grounding in a non-linguistic environment. Rather, the semantic context for interpretation, as well as the success criterion in evaluation is defined by successful execution of an action in the extrinsic environment, e.g., by receiving the correct answer from the database or by successful navigation to the destination. Recent attempts to learn semantic parsing from question-answer pairs without recurring to annotated logical forms have been presented by Kwiatowski et al. (2013), Berant et al. (2013), or Goldwasser and Roth (2013). The algorithms presented in these works are variants of structured prediction that take executability of semantic parses into account. Our work builds upon these ideas, however, to our knowledge the presented work is the first to embed translations into grounded scenarios in order to use feedback from interactions in these scenarios for structured learning in SMT.

A recent important research direction in SMT has focused on employing automated translation as an aid to human translators. *Computer assisted translation* (CAT) subsumes several modes of interaction, ranging from binary feedback on the quality of the system prediction (Saluja et al., 2012), to human post-editing operations on a system prediction resulting in a reference translation (Cesa-Bianchi et al., 2008), to human acceptance or overriding of sentence completion predictions (Langlais et al., 2000; Barrachina et al., 2008; Koehn and Haddow, 2009). In all interaction scenarios, it is important that the system learns dynamically from its errors in order to offer the user the experience of a system that adapts to the provided feedback. Since retraining the SMT model after each interaction is too costly, *online adaptation* after each interaction has become the learning protocol of choice for CAT. Online learning has been applied in generative SMT, e.g., using incremental versions of the EM algorithm (Ortiz-Martínez et al., 2010; Hardt and Elming, 2010), or in discriminative SMT, e.g., using perceptron-type algorithms (Cesa-Bianchi et al., 2008; Martínez-Gómez et al., 2012; Wäschle et al., 2013; Denkowski et al., 2014). In a similar way to deploying human feedback, extrinsic loss functions have been used to provide learning signals for SMT. For example, Nikoulina et al. (2012) propose a setup where an SMT system feeds into cross-language information retrieval, and receives feedback from the performance of translated queries with respect to cross-language retrieval performance. This feedback is used to train a reranker on an n-best list of translations order with respect to retrieval performance. In con-

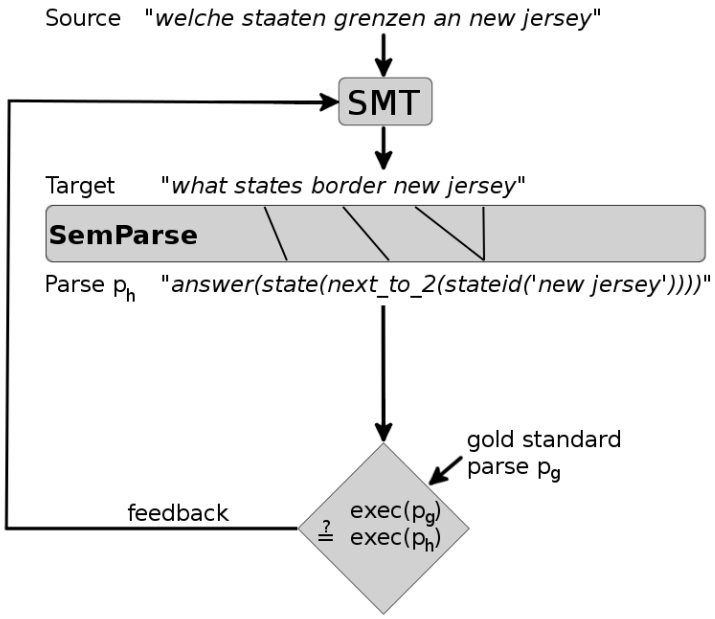


Figure 1: Response-based learning cycle for grounding SMT in virtual trivia gameplay.

trast to our work, all mentioned approaches to interactive or adaptive learning in SMT rely on human post-edits or human reference translations. Our work differs from these approaches in that exactly this dependency is alleviated by learning from responses in an extrinsic task.

Interactive scenarios have been used for evaluation purposes of translation systems for nearly 50 years, especially using *human reading comprehension* testing (Pfafflin, 1965; Fuji, 1999; Jones et al., 2005), and more recently, using face-to-face conversation mediated via machine translation (Sakamoto et al., 2013). However, despite offering direct and reliable prediction of translation quality, the cost and lack of reusability has confined task-based evaluations involving humans to *testing* scenarios, but prevented a use for interactive *training* of SMT systems as in our work.

Lastly, our work is related to *cross-lingual natural language processing* such as cross-lingual question answering or cross-lingual information retrieval as conducted at recent evaluation campaigns of the CLEF initiative.[1] While these approaches focus on improvements of the respective natural language processing task, our goal is to improve SMT by gathering feedback from the task.

[1] http://www.clef-initiative.eu

3 Grounding SMT in Semantic Parsing

In this paper, we present a proof-of-concept of our ideas of embedding SMT into simulated world environments as used in semantic parsing. We use the well-known GEOQUERY database on U.S. geography for this purpose. Embedding SMT in a semantic parsing scenario means to define translation quality by the ability of a semantic parser to construct a meaning representation from the translated query, which returns the correct answer when executed against the database. If viewed as simulated gameplay, a valid game move in this scenario returns the correct answer to a translated query.

The diagram in Figure 1 gives a sketch of response-based learning from semantic parsing in the geographical domain. Given a manual German translation of the English query as source sentence, the SMT system produces an English target translation. This sentence is fed into a semantic parser that produces an executable parse representation p_h. Feedback is generated by executing the parse against the database of geographical facts. Positive feedback means that the correct answer is received, i.e., $\text{exec}(p_g) \stackrel{?}{=} \text{exec}(p_h)$ indicates that the same answer is received from the gold standard parse p_g and the parse for the hypothesis translation p_h; negative feedback results in case a different or no answer is received.

The key advantage of response-based learning

is the possibility to receive positive feedback even from predictions that differ from gold standard reference translations, but yet receive the correct answer when parsed and matched against the database. Such structural and lexical variation broadens the learning capabilities in contrast to learning from fixed labeled data. For example, assume the following English query in the geographical domain, and assume positive feedback from executing the corresponding semantic parse against the geographical database:

```
Name prominent elevations in the
USA
```

The manual translation of the English original reads

```
Nenne prominente Erhebungen in
den USA
```

An automatic translation[2] of the German string produces the result

```
Give prominent surveys in the US
```

This translation will trigger negative task-based feedback: A comparison with the original allows the error to be traced back to the ambiguity of the German word `Erhebung`. Choosing a general domain translation instead of a translation appropriate for the geographical domain hinders the construction of a semantic parse that returns the correct answer from the database. An alternative translation might look as follows:

```
Give prominent heights in the US
```

Despite a large difference to the original English string, key terms such as `elevations` and `heights`, or `USA` and `US`, can be mapped into the same predicate in the semantic parse, thus allowing to receive positive feedback from parse execution against the geographical database.

4 Response-based Online Learning

Recent approaches to machine learning for SMT formalize the task of discriminating good from bad translations as a structured prediction problem. Assume a joint feature representation $\phi(x, y)$ of input sentences x and output translations $y \in Y(x)$, and a linear scoring function $s(x, y; w)$ for predicting a translation $\hat{y}$ (where $\langle \cdot, \cdot \rangle$ denotes the standard vector dot product) s.t.

$$\hat{y} = \underset{y \in Y(x)}{\arg\max}\ \mathrm{s}(x, y; w) = \underset{y \in Y(x)}{\arg\max} \langle w, \phi(x, y) \rangle .$$

[2] `http://translate.google.com`

The structured perceptron algorithm (Collins, 2002) learns an optimal weight vector w by updating w on input $x^{(i)}$ by the following rule, in case the predicted translation $\hat{y}$ is different from and scored higher than the reference translation $y^{(i)}$:

$$w = w + \phi(x^{(i)}, y^{(i)}) - \phi(x^{(i)}, \hat{y}).$$

This stochastic structural update aims to demote weights of features corresponding to incorrect decisions, and to promote weights of features for correct decisions.

An application of structured prediction to SMT involves more than a straightforward replacement of labeled output structures by reference translations. Firstly, update rules that require to compute a feature representation for the reference translation are suboptimal in SMT, because often human-generated reference translations cannot be generated by the SMT system. Such "unreachable" gold-standard translations need to be replaced by "surrogate" gold-standard translations that are close to the human-generated translations and still lie within the reach of the SMT system. Computation of distance to the reference translation usually involves cost functions based on sentence-level BLEU (Nakov et al. (2012), *inter alia*) and incorporates the current model score, leading to various ramp loss objectives described in Gimpel and Smith (2012).

An alternative approach to alleviate the dependency on labeled training data is response-based learning. Clarke et al. (2010) or Goldwasser and Roth (2013) describe a response-driven learning framework for the area of semantic parsing: Here a meaning representation is "tried out" by iteratively generating system outputs, receiving feedback from world interaction, and updating the model parameters. Applied to SMT, this means that we predict translations and use positive response from acting in the world to create "surrogate" gold-standard translations. This decreases the dependency on a few (mostly only one) reference translations and guides the learner to promote translations that perform well with respect to the extrinsic task.

In the following, we will present a framework that combines standard structured learning from given reference translations with response-based learning from task-approved references. We need to ensure that gold-standard translations lead to positive task-based feedback, that means they can

be parsed and executed successfully against the database. In addition, we can use translation-specific cost functions based on sentence-level BLEU in order to boost similarity of translations to human reference translations.

We denote feedback by a binary execution function $\mathrm{e}(y) \in \{1, 0\}$ that tests whether executing the semantic parse for the prediction against the database receives the same answer as the parse for the gold standard reference. Our cost function $\mathrm{c}(y^{(i)}, y) = (1 - \mathrm{BLEU}(y^{(i)}, y))$ is based on a version of sentence-level BLEU Nakov et al. (2012). Define y^+ as a surrogate gold-standard translation that receives positive feedback, has a high model score, and a low cost of predicting y instead of $y^{(i)}$:

$$y^+ = \underset{y \in Y(x^{(i)}):\mathrm{e}(y)=1}{\arg\max} \Big(\mathrm{s}(x^{(i)}, y; w) - \mathrm{c}(y^{(i)}, y) \Big) .$$

The opposite of y^+ is the translation y^- that leads to negative feedback, has a high model score, and a high cost. It is defined as follows:

$$y^- = \underset{y \in Y(x^{(i)}):\mathrm{e}(y)=0}{\arg\max} \Big(\mathrm{s}(x^{(i)}, y; w) + \mathrm{c}(y^{(i)}, y) \Big) .$$

Update rules can be derived by minimization of the following ramp loss objective:

$$\min_w \Bigg(- \max_{y \in Y(x^{(i)}):\mathrm{e}(y)=1} \Big(\mathrm{s}(x^{(i)}, y; w) - \mathrm{c}(y^{(i)}, y) \Big) + \max_{y \in Y(x^{(i)}):\mathrm{e}(y)=0} \Big(\mathrm{s}(x^{(i)}, y; w) + \mathrm{c}(y^{(i)}, y) \Big) \Bigg) .$$

Minimization of this objective using stochastic (sub)gradient descent (McAllester and Keshet, 2011) yields the following update rule:

$$w = w + \phi(x^{(i)}, y^+) - \phi(x^{(i)}, y^-).$$

The intuition behind this update rule is to discriminate the translation y^+ that leads to positive feedback and best approximates (or is identical to) the reference within the means of the model from a translation y^- which is favored by the model but does not execute and has high cost. This is done by putting all the weight on the former.

Algorithm 1 presents pseudo-code for our response-driven learning scenario. Upon predicting translation $\hat{y}$, in case of positive feedback from the task, we treat the prediction as surrogate reference by setting $y^+ \leftarrow \hat{y}$, and by adding it to the set of reference translations for future use. Then we need to compute y^-, and update by the difference in feature representations of y^+ and y^-, at a learning rate η. If the feedback is negative, we want to move the weights away from the prediction, thus we treat it as y^-. To perform an update, we need to compute y^+. If either y^+ or y^- cannot be computed, the example is skipped.

Algorithm 1 Response-based Online Learning

```
repeat
    for i = 1, ..., n do
        Receive input string x^(i)
        Predict translation ŷ
        Receive task feedback e(ŷ) ∈ {1, 0}
        if e(ŷ) = 1 then
            y^+ ← ŷ
            Store ŷ as reference y^(i) for x^(i)
            Compute y^-
        else
            y^- ← ŷ
            Receive reference y^(i)
            Compute y^+
        end if
        w ← w + η(φ(x^(i), y^+) − φ(x^(i), y^-))
    end for
until Convergence
```

The sketched algorithm allows several variations. In the form depicted above, it allows to use human reference translations in addition to task-approved surrogate references. The cost function can be implemented by different versions of sentence-wise BLEU, or it can be omitted completely so that learning relies on task-based feedback alone, similar to algorithms recently suggested for semantic parsing (Goldwasser and Roth, 2013; Kwiatowski et al., 2013; Berant et al., 2013). Lastly, regularization can be introduced by using update rules corresponding to primal form optimization variants of support vector machines (Collobert and Bengio, 2004; Chapelle, 2007; Shalev-Shwartz et al., 2007).

5 Experiments

5.1 Experimental Setup

In our experiments, we use the GEOQUERY database on U.S. geography as provided by Jones

method	precision	recall	F1	BLEU
1 CDEC	63.67	58.21	60.82	46.53
2 EXEC	70.36	63.57	66.79[1]	48.00[1]
3 RAMPION	75.58	69.64	72.49[12]	**56.64**[12]
4 REBOL	**81.15**	**75.36**	**78.15**[123]	55.66[12]

Table 1: Experimental results using extended parser for returning answers from GEOQUERY (precision, recall, F1) and n-gram match to original English query (BLEU) on 280 re-translated test examples. Best results for each column are highlighted in **bold face**. Superscripts [1234] denote a significant improvement over the respective method.

method	precision	recall	F1	BLEU
1 CDEC	65.59	57.86	61.48	46.53
2 EXEC	66.54	61.79	64.07	46.00
3 RAMPION	67.68	63.57	65.56	**55.67**[12]
4 REBOL	**70.68**	**67.14**	**68.86**[12]	**55.67**[12]

Table 2: Experimental results using the original parser for returning answers from GEOQUERY (precision, recall, F1) and n-gram match to original English query (BLEU) on 280 re-translated test examples.

et al. (2012).[3] The dataset includes 880 English questions and their logical forms. The English strings were manually translated into German by the authors of Jones et al. (2012)), and corrected for typos by the authors of this paper. We follow the provided split into 600 training examples and 280 test examples.

For response-based learning, we retrained the semantic parser of Andreas et al. (2013)[4] on the full 880 GEOQUERY examples in order to reach full parse coverage. This parser is itself based on SMT, trained on parallel data consisting of English queries and linearized logical forms, and on a language model trained on linearized logical forms. We used the hierarchical phrase-based variant of the parser. Note that we do not use GEOQUERY test data in SMT training. Parser training includes GEOQUERY test data in order to be less dependent on parse and execution failures in the evaluation: If a translation system, response-based or reference-based, translates the German input into the gold standard English query it should be rewarded by positive task feedback. To double-check whether including the 280 test examples in parser training gives an unfair advantage to response-based learning, we also present experimental results using the original parser of Andreas et al. (2013) that is trained only on the 600 GEOQUERY training examples.

The bilingual SMT system used in our experiments is the state-of-the-art SCFG decoder CDEC (Dyer et al., 2010)[5]. We built grammars using its implementation of the suffix array extraction method described in Lopez (2007). For language modeling, we built a modified Kneser-Ney smoothed 5-gram language model using the English side of the training data. We trained the SMT system on the English-German parallel web data provided in the COMMON CRAWL[6] (Smith et al., 2013) dataset.

5.2 Compared Systems

Method 1 is the baseline system, consisting of the CDEC SMT system trained on the COMMON CRAWL data as described above. This system does not use any GEOQUERY data for training. Methods 2-4 use the 600 training examples from GEOQUERY for discriminative training only.

Variants of the response-based learning algorithm described above are implemented as a standalone tool that operates on CDEC n-best lists of 10,000 translations of the GEOQUERY training data. All variants use sparse features of CDEC as described in Simianer et al. (2012) that extract rule

[3] http://homepages.inf.ed.ac.uk/s1051107/geoquery-2012-08-27.zip

[4] https://github.com/jacobandreas/smt-semparse

[5] https://github.com/redpony/cdec

[6] http://www.statmt.org/wmt13/training-parallel-commoncrawl.tgz

prediction:	how many inhabitants has new york
reference:	how many people live in new york
prediction:	how big is the population of texas
reference:	how many people live in texas
prediction:	which are the cities of the state with the highest elevation
reference:	what are the cities of the state with the highest point
prediction:	how big is the population of states , through which the mississippi runs
reference:	what are the populations of the states through which the mississippi river runs
prediction:	what state borders california
reference:	what is the adjacent state of california
prediction:	what are the capitals of the states which have cities with the name durham
reference:	what is the capital of states that have cities named durham
prediction:	what rivers go through states with the least cities
reference:	which rivers run through states with fewest cities

Table 3: Predicted translations by response-based learning (REBOL) leading to positive feedback versus gold standard references.

shapes, rule identifiers, and bigrams in rule source and target directly from grammar rules. Method 4, named REBOL, implements REsponse-Based Online Learning by instantiating y^+ and y^- to the form described in Section 4: In addition to the model score s, it uses a cost function c based on sentence-level BLEU (Nakov et al., 2012) and tests translation hypotheses for task-based feedback using a binary execution function e. This algorithm can convert predicted translations into references by task-feedback, and additionally use the given original English queries as references. Method 2, named EXEC, relies on task-execution by function e and searches for executable or non-executable translations with highest score s to distinguish positive from negative training examples. It does not use a cost function and thus cannot make use of the original English queries.

We compare response-based learning with a standard structured prediction setup that omits the use of the execution function e in the definition of y^+ and y^-. This algorithm can be seen as a stochastic (sub)gradient descent variant of RAMPION (Gimpel and Smith, 2012). It does not make use of the semantic parser, but defines positive and negative examples based on score s and cost c with respect to human reference translations.

We report BLEU (Papineni et al., 2001) of translation system output measured against the original English queries. Furthermore, we report precision, recall, and F1-score for executing semantic parses built from translation system outputs against the GEOQUERY database. Precision is defined as the percentage of correctly answered examples out of those for which a parse could be produced; recall is defined as the percentage of total examples answered correctly; F1-score is the harmonic mean of both. Statistical significance is measured using Approximate Randomization (Noreen, 1989) where result differences with a p-value smaller than 0.05 are considered statistically significant.

Methods 2-4 perform structured learning for SMT on the 600 GEOQUERY training examples and re-translate the 280 unseen GEOQUERY test data, following the data split of Jones et al. (2012). Training for RAMPION, REBOL and EXEC was repeated for 10 epochs. The learning rate η is set to a constant that is adjusted by cross-validation on the 600 training examples.

5.3 Empirical Results

We present an experimental comparison of the four different systems according to BLEU and

reference	RAMPION	REBOL
how many colorado rivers are there	how many rivers with the name colorado gives it	how many rivers named colorado are there
what are the populations of states which border texas	how big are the populations of the states , which in texas borders	how big are the populations of the states which on texas border
what is the biggest capital city in the us	what is the largest city in the usa	what is the largest capital in the usa
what state borders new york	what states limits of new york	what states border new york
which states border the state with the smallest area	what states boundaries of the state with the smallest surface area	what states border the state with the smallest surface area

Table 4: Predicted translations by response-based learning (REBOL) leading to positive feedback versus translations by supervised structured learning (RAMPION) leading to negative feedback.

F1, using an extended semantic parser (trained on 880 GEOQUERY examples) and the original parser (trained on 600 GEOQUERY training examples). The extended parser reaches and F1-score of 99.64% on the 280 GEOQUERY test examples; the original parser yields an F1-score of 82.76%.

Table 1 reports results for the extended semantic parser. A system ranking according to F1-score shows about 6 points difference between the respective methods, ranking REBOL over RAMPION, EXEC and CDEC. The exploitation of task-feedback allows both EXEC and REBOL to improve task-performance over the baseline. REBOL's combination of task feedback with a cost function achieves the best results since positively executable hypotheses and reference translations can both be exploited to guide the learning process. Since all English reference queries lead to positively executable parses in the setup that uses the extended semantic parser, RAMPION implicitly also has access to task feedback. This allows RAMPION to improve F1 over the baseline. All result differences are statistically significant.

In terms of BLEU score measured against the original English GEOQUERY queries, the best nominal result is obtained by RAMPION which uses them as reference translations. REBOL performs worse since BLEU performance is optimized only implicitly in cases where original English queries function as positive examples. However, the result differences between these two systems do not score as statistically significant. Despite not optimizing for BLEU performance against references, the fact that positively executable translations include the references allows even EXEC to improve BLEU over CDEC which does not use GEOQUERY data at all in training. This result difference is statistically significant.

Table 2 compares the same systems using the original parser trained on 600 training examples. The system ranking according to F1-score shows the same ordering that is obtained when using an extended semantic parser. However, the respective methods are separated only by 3 or less points in F1 score such that only the result difference of REBOL over the baseline CDEC and over EXEC is statistically significant. We conjecture that this is due to a higher number of empty parses on the test set which makes this comparison unstable.

In terms of BLEU measured against the original queries, the result differences between REBOL and RAMPION are not statistically significant, and neither are the result differences between EXEC and CDEC. The result differences between systems of the former group and the systems of latter group are statistically significant.

5.4 Error Analysis

For a better understanding of the differences between the results produced by supervised and response-based learning, we conducted an er-

reference	RAMPION	REBOL
how many states have a higher point than the highest point of the state with the largest capital city in the us	how many states have a higher nearby point as the highest point of the state with the largest capital in the usa	how many states have a high point than the highest point of the state with the largest capital in the usa
how tall is mount mckinley	how high is mount mckinley	what is mount mckinley
what is the longest river that flows through a state that borders indiana	how is the longest river , which runs through a state , borders the of indiana	what is the longest river which runs through a state of indiana borders
what states does the mississippi river run through	through which states runs the mississippi	through which states is the mississippi
which is the highest peak not in alaska	how is the highest peaks of not in alaska is	what is the highest peak in alaska is

Table 5: Predicted translations where supervised structured learning (RAMPION) leads to positive feedback versus translations by response-based learning (REBOL) leading to negative feedback.

ror analysis on the test examples. Table 3 shows examples where the translation predicted by response-based learning (REBOL) differs from the gold standard reference translation, but yet leads to positive feedback via a parse that returns the correct answer from the database. The examples show structural and lexical variation that leads to differences on the string level at equivalent positive feedback from the extrinsic task. This can explain the success of response-based learning: Lexical and structural variants of reference translations can be used to boost model parameters towards translations with positive feedback, while the same translations might be considered as negative examples in standard structured learning.

Table 4 shows examples where translations from REBOL and RAMPION differ from the gold standard reference, and predictions by REBOL lead to positive feedback, while predictions by RAMPION lead to negative feedback. Table 5 shows examples where translations from RAMPION outperform translations from REBOL in terms of task feedback. We see that predictions from both systems are in general grammatical. This can be attributed to the use of sentence-level BLEU as cost function in RAMPION and REBOL. Translation errors of RAMPION can be traced back to mistranslations of key terms (`city` versus `capital`, `limits` or `boundaries` versus `border`). Translation errors of REBOL more frequently show missing translations of terms.

6 Conclusion

We presented a proposal for a new learning and evaluation framework for SMT. The central idea is to ground meaning transfer in successful interaction in an extrinsic task, and use task-based feedback for structured learning. We presented a proof-of-concept experiment that defines the extrinsic task as executing semantic parses of translated queries against the GEOQUERY database. Our experiments show an improvement of about 6 points in F1-score for response-based learning over structured learning from reference translations. Our error analysis shows that response-based learning generates grammatical translations which is due to the additional use of a cost function that boosts similarity of translations to human reference translations.

In future work, we would like to extend our work on embedding SMT in virtual gameplay to larger and more diverse datasets, and involve human feedback in the response-based learning loop.

References

Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic parsing as machine translation. In

Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL'13), Sofia, Bulgaria.

Sergio Barrachina, Oliver Bender, Francisco Casacuberta, Jorge Civera, Elsa Cubel, Shahram Khadivi, Antonio Lagarda, Hermann Ney, Jesús Tomás, Enrique Vidal, and Juan-Miguel Vilar. 2008. Statistical approaches to computer-assisted translation. *Computational Linguistics*, 35(1):3–28.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP'13)*, Seattle, WA.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extenstion. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL'13)*, Sofia, Bulgaria.

Nicolò Cesa-Bianchi, Gabriele Reverberi, and Sandor Szedmak. 2008. Online learning algorithms for computer-assisted translation. Technical report, SMART (`www.smart-project.eu`).

Olivier Chapelle. 2007. Training a support vector machine in the primal. *Neural Computation*, 19(5):1155–1178.

David L. Chen and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI'11)*, pages 859–866, San Francisco, CA.

James Clarke, Dan Goldwasser, Wing-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of the 14th Conference on Natural Language Learning (CoNLL'10)*, pages 18–27, Uppsala, Sweden.

Michael Collins. 2002. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *Proceedings of the conference on Empirical Methods in Natural Language Processing (EMNLP'02)*, Philadelphia, PA.

Ronan Collobert and Samy Bengio. 2004. Links between perceptrons, MLPs, and SVMs. In *Proceedings of the 21st International Conference on Machine Learning (ICML'04)*, Banff, Canada.

Michael Denkowski, Chris Dyer, and Alon Lavie. 2014. Learning from post-editing: Online model adaptation for statistical machine translation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL'14)*, Gothenburg, Sweden.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Jonathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the ACL 2010 System Demonstrations*, Uppsala, Sweden.

Masaru Fuji. 1999. Evaluation experiment for reading comprehension of machine translation outputs. In *Proceedings of the Machine Translation Summit VII*, Singapore.

Kevin Gimpel and Noah A. Smith. 2012. Structured ramp loss minimization for machine translation. In *Proceedings of 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2012)*, Montreal, Canada.

Dan Goldwasser and Dan Roth. 2013. Learning from natural instructions. *Machine Learning*, 94(2):205–232.

Daniel Hardt and Jakob Elming. 2010. Incremental re-training for post-editing SMT. In *Proceedings of the 9th Conference of the Association for Machine Tranlation in the Americas (AMTA'10)*, Denver, CO.

Douglas Jones, Wade Shen, Neil Granoien, Martha Herzog, and Clifford Weinstein. 2005. Measuring translation quality by testing english speakers with a new defense language proficiency test for arabic. In *Proceedings of 2005 International Conference on Intelligence Analysis*, McLean, VA.

Bevan K. Jones, Mark Johnson, and Sharon Goldwater. 2012. Semantic parsing with bayesion tree transducers. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL'12)*, Jeju Island, Korea.

Philipp Koehn and Barry Haddow. 2009. Interactive assistance to human translators using statistical machine translation methods. In *Proceedings of MT Summit XII*, Ottawa, Ontario, Canada.

Tom Kwiatowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP'13)*, Seattle, WA.

Philippe Langlais, George Foster, and Guy Lapalme. 2000. Transtype: a computer-aided translation typing system. In *Proceedings of the ANLP-NAACL 2000 Workshop on Embedded Machine Translation Systems*, Seattle, WA.

Adam Lopez. 2007. Hierarchical phrase-based translation with suffix arrays. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, Prague, Czech Republic.

Pascual Martínez-Gómez, Germán Sanchis-Trilles, and Francisco Casacuberta. 2012. Online adaptation

strategies for statistical machine translation in post-editing scenarios. *Pattern Recognition*, 45(9):3193–3202.

David McAllester and Joseph Keshet. 2011. Generalization bounds and consistency for latent structural probit and ramp loss. In *Proceedings of the 25th Annual Conference on Neural Information Processing Sytems (NIPS 2011)*, Granada, Spain.

Preslav Nakov, Francisco Guzmán, and Stephan Vogel. 2012. Optimizing for sentence-level bleu+1 yields short translations. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*, Bombay, India.

Vassilina Nikoulina, Bogomil Kovachev, Nikolaos Lagos, and Christof Monz. 2012. Adaptation of statistical machine translation model for cross-lingual information retrieval in a service context. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL'12)*, Avignon, France.

Eric W. Noreen. 1989. *Computer Intensive Methods for Testing Hypotheses. An Introduction*. Wiley, New York.

Daniel Ortiz-Martínez, Ismal García-Varea, and Francisco Casacuberta. 2010. Online learning for interactive statistical machine translation. In *Proceedings of the Human Language Technologies conference and the 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL'10)*, Los Angeles, CA.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. Technical Report IBM Research Division Technical Report, RC22176 (W0190-022), Yorktown Heights, N.Y.

Sheila M. Pfafflin. 1965. Evaluation of machine translations by reading comprehension tests and subjective judgements. *Mechanical Translation and Computational Linguistics*, 8(2):2–8.

Akiko Sakamoto, Nayuko Watanabe, Satoshi Kamatani, and Kazuo Sumita. 2013. Development of a simultaneous interpretation system for face-to-face services and its evaluation experiment in real situation. In *Proceedings of the Machine Translation Summit XIV*, Nice, France.

Avneesh Saluja, Ian Lane, and Ying Zhang. 2012. Machine translation with binary feedback: A large-margin approach. In *Proceedings of the 10th Biennial Conference of the Association for Machine Translation in the Americas (AMTA'12)*, San Diego, CA.

Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. 2007. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In *Proceedings of the 24th International Conference on Machine Learning (ICML'07)*, Corvallis, OR.

Patrick Simianer, Stefan Riezler, and Chris Dyer. 2012. Joint feature selection in distributed stochastic learning for large-scale discriminative training in SMT. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012)*, Jeju, Korea.

Jason R. Smith, Herve Saint-Amand, Magdalena Plamada, Philipp Koehn, Chris Callison-Burch, and Adam Lopez. 2013. Dirt cheap web-scale parallel text from the common crawl. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL'13)*, Sofia, Bulgaria.

Katharina Wäschle, Patrick Simianer, Nicola Bertoldi, Stefan Riezler, and Marcello Federico. 2013. Generative and discriminative methods for online adaptation in SMT. In *Proceedings of the Machine Translation Summit XIV*, Nice, France.

Yuk Wah Wong and Raymond J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL'06)*, New York City, NY.

Luke S. Zettlemoyer and Michael Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL-IJCNLP'09)*, Singapore.

Modelling Events through Memory-based, Open-IE Patterns for Abstractive Summarization

Daniele Pighin
Google Inc.
biondo@google.com

Marco Cornolti*
University of Pisa, Italy
cornolti@di.unipi.it

Enrique Alfonseca
Google Inc.
ealfonseca@google.com

Katja Filippova
Google Inc.
katjaf@google.com

Abstract

Abstractive text summarization of news requires a way of representing events, such as a collection of pattern clusters in which every cluster represents an event (e.g., *marriage*) and every pattern in the cluster is a way of expressing the event (e.g., *X married Y, X and Y tied the knot*). We compare three ways of extracting event patterns: heuristics-based, compression-based and memory-based. While the former has been used previously in multi-document abstraction, the latter two have never been used for this task. Compared with the first two techniques, the memory-based method allows for generating significantly more grammatical and informative sentences, at the cost of searching a vast space of hundreds of millions of parse trees of known grammatical utterances. To this end, we introduce a data structure and a search method that make it possible to efficiently extrapolate from every sentence the parse sub-trees that match against any of the stored utterances.

1 Introduction

Text summarization beyond extraction requires a semantic representation that abstracts away from words and phrases and from which a summary can be generated (Mani, 2001; Spärck-Jones, 2007). Following and extending recent work in semantic parsing, information extraction (IE), paraphrase generation and summarization (Titov and Klementiev, 2011; Alfonseca et al., 2013; Zhang and Weld, 2013; Mehdad et al., 2013), the representation we consider in this paper is a large collection of clusters of event patterns. An abstractive summarization system relying on such a representation proceeds by (1) detecting the most relevant event cluster for a given sentence or sentence collection, and (2) using the most representative pattern from the cluster to generate a concise summary sentence. Figure 1 illustrates the summarization architecture we are assuming in this paper. Given input text(s) with resolved and typed entity mentions, event mentions and the most relevant event cluster are detected (first arrow). Then, a summary sentence is generated from the event and entity representations (second arrow).

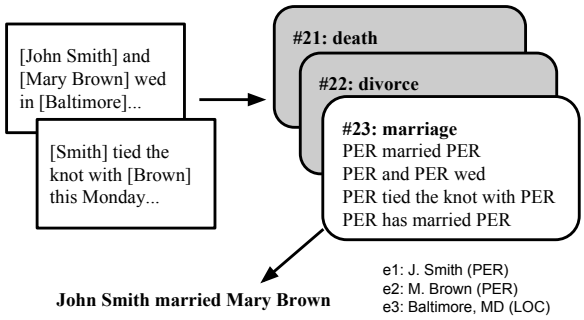


Figure 1: An example of abstracting from input sentences to an event representation and generation from that representation.

However, the utility of such a representation for summarization depends on the quality of pattern clusters. In particular, event patterns must correspond to grammatically correct sentences. Introducing an incomplete or incomprehensible pattern (e.g., *PER said PER*) may negatively affect both event detection and sentence generation. Related work on paraphrase detection and relation extraction is mostly *heuristics-based* and has relied on hand-crafted rules to collect such patterns (see Sec. 2). A standard approach is to focus on binary relations between entities and extract

*Work done during an internship at Google Zurich.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 892–901,
Baltimore, Maryland, USA, June 23-25 2014.

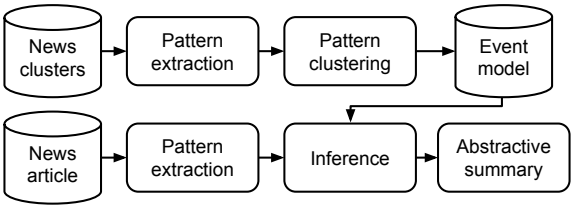


Figure 2: A generic pipeline for event-driven abstractive headline generation.

the dependency path between the two entities as an event representation. An obvious limitation of this approach is there is no guarantee that the extracted pattern corresponds to a grammatically correct sentence, e.g., that an essential prepositional phrase is retained like in *file for a divorce*.

In this paper we explore two novel, data-driven methods for event pattern extraction. The first, *compression-based* method uses a robust sentence compressor with an aggressive compression rate to get to the core of the sentence (Sec. 3). The second, *memory-based* method relies on a vast collection of human-written headlines and sentences to find a substructure which is known to be grammatically correct (Sec. 4). While the latter method comes closer to ensuring perfect grammaticality, it introduces a problem of efficiently searching the vast space of known well-formed patterns. Since standard iterative approaches comparing every pattern with every sentence are prohibitive here, we present a search strategy which scales well to huge collections (hundreds of millions) of sentences.

In order to evaluate the three methods, we consider an abstractive summarization task where the goal is to get the gist of single sentences by recognizing the underlying event and generating a short summary sentence. To the best of our knowledge, this is the first time that this task has been proposed; it can be considered as *abstractive sentence compression*, in contrast to most existing sentence compression systems which are based on selecting words from the original sentence or rewriting with simpler paraphrase tables. An extensive evaluation with human raters demonstrates the utility of the new pattern extraction techniques. Our analysis highlights advantages and disadvantages of the three methods.

To better isolate the qualities of the three extraction methodologies, all three methods use the same training data and share components of the very same summarization architecture, as shown in Figure 2: an event model is constructed by clustering the patterns extracted according to the selected extraction method. Then, the same extraction method is used to collect patterns from sentences in never-seen-before news articles. Finally, the patterns are used to query the event model and generate an abstractive summary. The three different pattern extractors are detailed in the next three sections.

Algorithm 1 HEURISTICEXTRACTOR(T, E): heuristically extract relational patterns for the dependency parse T and the set of entities E.

```
/* Global constants /*
global V_p, V_c, N_p, N_c
V_c ← {subj, nsubj, nsubjpass, dobj, iobj, xcomp,
       acomp, expl, neg, aux, attr, prt}
V_p ← {xcomp}
N_c ← {det, predet, num, ps, poss, nc, conj}
N_p ← {ps, poss, subj, nsubj, nsubjpass, dobj, iobj}
/* Entry point /*
P ← ∅
for all C ∈ COMBINATIONS(E) do
  N ← MENTIONNODES(T, C)
  N' ← APPLYHEURISTICS(T, BUILDMST(T, N))
  P ← P ∪ {BUILDPATTERN(T, N')}
return P
/* Procedures /*
procedure APPLYHEURISTICS(T, N)
  N' ← N
  while |N'| > 0 do
    N'' ← ∅
    for all n ∈ N' do
      if n.ISVERB() then
        N'' ← N'' ∪ INCLUDECHILDREN(n, V_c)
        N'' ← N'' ∪ INCLUDEPARENT(n, V_p)
      else if n.ISNOUN() then
        N'' ← N'' ∪ INCLUDECHILDREN(n, N_c)
        N'' ← N'' ∪ INCLUDEPARENT(n, N_p)
    N' ← N'' \ N'
procedure INCLUDECHILDREN(n, L)
  R ← ∅
  for all c ∈ n.CHILDREN() do
    if c.PARENTEDGELABEL() ∈ L then
      R ← R ∪ {c}
  return R
procedure INCLUDEPARENT(n, L)
  if n.PARENTEDGELABEL() ∈ L then
    return {n}
  else return ∅
```

2 Heuristics-based pattern extraction

In order to be able to work in an Open-IE manner, applicable to different domains, most existing pattern extraction systems are based on linguistically motivated heuristics. Zhang and Weld (2013) is based on REVERB (Fader et al., 2011), which uses a regular expression on part-of-speech tags to produce the extractions. An alternative system,

OLLIE (Schmitz et al., 2012), uses syntactic dependency templates to guide the pattern extraction process.

The heuristics used in this paper are inspired by Alfonseca et al. (2013), who built well formed relational patterns by extending minimum spanning trees (MST) which connect entity mentions in a dependency parse. Algorithm 1 details our reimplementation of their method and the specific set of rules that we rely on to enforce pattern grammaticality. We use the standard Stanford-style set of dependency labels (de Marneffe et al., 2006). The input to the algorithm are a parse tree T and a set of target entities E. We first generate combinations of 1-3 elements of E (line 10), then for each combination C we identify all the nodes in T that mention any of the entities in C. We continue by constructing the MST of these nodes, and finally apply our heuristics to the nodes in the MST. The procedure APPLYHEURISTICS (:16) recursively grows a nodeset N' by including children and parents of noun and verb nodes in N' based on dependency labels. For example, we include all children of verbs in N' whose label is listed in V_c (:3), e.g., active or passive subjects, direct or indirect objects, particles and auxiliary verbs. Similarly, we include the parent of a noun in N' if the dependency relation between the node and its parent is listed in N_p.

3 Pattern extraction by sentence compression

Sentence compression is a summarization technique that shortens input sentences preserving the most important content (Grefenstette, 1998; McDonald, 2006; Clarke and Lapata, 2008, inter alia). While first attempts at integrating a compression module into an extractive summarization system were not particularly successful (Daumé III and Marcu, 2004, inter alia), recent work has been very promising (Berg-Kirkpatrick et al., 2011; Wang et al., 2013). It has shown that dropping constituents of secondary importance from selected sentences – e.g., temporal modifiers or relative clauses – results in readable and more informative summaries. Unlike this related work, our goal here is to compress sentences to obtain an event pattern – the minimal grammatical structure expressing an event. To our knowledge, this application of sentence compressors is novel. As in Section 2, we only consider sentences mentioning entities and require the compression (pattern) to retain at least one such mention.

Sentence compression methods are abundant but very few can be configured to produce output satisfying certain constraints. For example, most compression algorithms do not accept compression rate as an argument. In our case, sentence compressors which formulate the compression task as an optimization problem and solve it with integer linear programming (ILP) tools under a number of constraints are particularly attractive (Clarke and Lapata, 2008; Filippova and Altun, 2013). They can be extended relatively easily with both the length constraint and the constraint on retaining certain words. The method of Clarke and Lapata (2008) uses a trigram language model (LM) to score compressions. Since we are interested in very short outputs, a LM trained on standard, uncompressed text would not be suitable. Instead, we chose to modify the method of Filippova and Altun (2013) because it relies on dependency parse trees and does not use any LM scoring.

Like other syntax-based compressors, the system of Filippova and Altun (2013) prunes dependency structures to obtain compression trees and hence sentences. The objective function to maximize in an ILP problem (Eq. 1) is formulated over weighted edges in a transformed dependency tree and is subject to a number of constraints. Edge weight is defined as a linear function over a feature set: $w(e) = \mathbf{w} \cdot \mathbf{f}(e)$.

$$F(X) = \sum_{e \in E} x_e \times w(e) \quad (1)$$

In our reimplementation we followed the algorithm as described by Filippova and Altun (2013). The compression tree is obtained in two steps. First, the input tree is transformed with deterministic rules, most of which aim at collapsing indispensable modifiers with their heads (determiners, auxiliary verbs, negation, multi-word expressions, etc.). Then a sub-tree maximizing the objective function is found under a number of constraints.

Apart from the structural constrains from the original system which ensure that the output is a valid tree, the constraints we add state that:

1. tree size in edges must be in [3, 6],
2. entity mentions must be retained,
3. subject of the clause must be retained,
4. the sub-tree must be covered by a single clause – exactly one finite verb must used.

Since we consider compressions with different lengths as candidates, from this set we select the one with the maximum averaged edge weight as the final compression. Figure 3 illustrates the use of the compressor for obtaining event patterns. Dashed edges are dropped as a result of constrained compression so that the output is *John Smith married Mary Brown* and the event pattern is *PER married PER*. Note that the root of a subclause is allowed to be the top-level node in the extracted compression.

Compared with patterns obtaines with heuristics, compression patterns should retain prepositional verb arguments whose removal would render the pattern ungrammatical. As an example consider *[C. Zeta-Jones] and [M. Douglas] filed for divorce*. The heuristics-based pattern is *PER and PER filed* which is incomplete. Unlike it, the compression-based method keeps the essential prepositional phrase *for divorce* in the pattern because the average edge weight is greater for the tree with the prepositional phrase.

4 Memory-based pattern extraction

Neither heuristics-based, nor compression-based methods provide a guarantee that the extracted pattern is grammatically correct. In this section we introduce an extraction technique which makes it considerably more likely because it only extracts patterns which have been observed as full sentences in a human-written text (Sec. 4.1). However, this memory-based method also poses a problem not encountered by the two previous methods: how to search over the vast space of observed headlines and sentences to extract a pattern from a given sentence? Our trie-based solution, which we present in the remainder of this section, makes it possible to compare a dependency graph against millions of observed grammatical utterances in a fraction of a second.

4.1 A *tree-trie* to store them all...

Our objective is to construct a compact representation of hundreds of millions of observed sentences that can fit in the memory of a standard workstation. This data structure should make it possible to efficiently identify the sub-trees of a sentence that match any complete utterance previously observed. To this end, we build a trie of dependency trees (which we call a *tree-trie*) by scanning all the dependency parses in the news training data, and index each tree in the tree-trie according to Algorithm 2. For better clarity, the process is also described graphically in Figure 4. First, each dependency tree (a) is *linearized*, resulting in a data structure that consists of two aligned sequences (b). The first sequence (*tokens*) encodes word/parent-relation pairs, while the second sequence (*structure*) encodes the offsets of parent nodes in the linearized tree. As an example, the first word "The" is a determiner ("det") for the second node (offset 1) in the sequence, which is "cat". In turn, "cat" is the subject ("nsubj") of the node in position 2, i.e., "sleeps". As described in Algorithm 2, we recursively store the token sequence in the trie, each word/relation pair being stored in a node. When the token sequence is completely consumed, we store in the current trie node the structure of the linearized tree. Combining structural information with the sequential information encoded by each path in the trie makes it possible to rebuild a complete dependency graph. Figure 4(c) shows an example trie encoding 4 different sentences. We highlighted in bold the path corresponding to the linearized form (b) of the example parse tree (a).

Algorithm 2 STORE(T, I): store the dependency tree T in the tree-trie I.

```
1:  /* Entry point */
2:  L ← T.LINEARIZE()
3:  STORERECURSION(I.ROOT(), L, 0)
4:  return M
5:  /* Procedures */
6:  procedure STORERECURSION(n, L, o)
7:    if o == L.LENGTH() then
8:      n.ADDTREESTRUCTURE(L.STRUCTURE())
9:      return
10:   if not n.HASCHILD(L.TOKEN(o)) then
11:     n.ADDCHILD(L.TOKEN(o))
12:   n' ← n.GETCHILD(L.TOKEN(o))
13:   STORERECURSION(n', L, o + 1)
```

The figure shows that the tree contains two kinds of nodes: end-of-sentence (EOS) nodes (red) and non-terminal nodes (in blue). EOS nodes do not necessarily coincide with trie leaves, as it is possible to observe complete sentences embedded in longer ones. EOS nodes differ from non-terminal nodes in that they store one or more structural sequences corresponding to different syntactic representations of observed sentences with the same tokens.

Space-complexity and generalization. Storing all the observed sentences in a single trie requires huge amounts of memory. To make it possible to

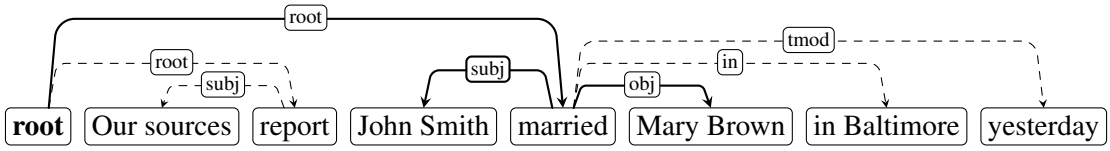


Figure 3: Transformed dependency tree with a sub-tree expressing an event pattern.

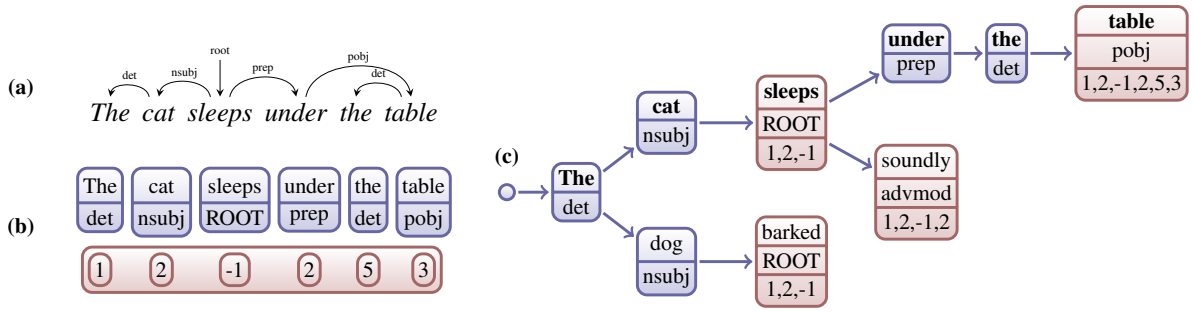


Figure 4: A dependency tree (a), its linearized form (b) and the resulting path in a trie (c), in bold.

store a complete tree-trie in memory, we adopt the following strategy. We replace the surface form of entity nodes with the coarse entity type (e.g., PER, LOC, ORG) of the entity. Similarly, we replace proper nouns with the placeholder "[P]", thus significantly reducing lexical sparsity. Then, we encode each distinct word/relation pair as a 32-bit unsigned integer. Assuming a maximum tree size of 255 nodes, we represent structure sequences as vectors of type unsigned char (8 bit per element). Finally, we store trie-node children as sorted vectors instead of hash maps to reduce memory footprint. As a result, we are able to load a trie encoding 400M input dependency parses, 170M distinct nodes and 48M distinct sentence structures in under 10GB of RAM.

4.2 . . . and in the vastness match them

At lookup time, we want to use the tree-trie to identify all sub-graphs of an input dependency tree T that match at least a complete observed sentence. To do so, we need to identify all paths in the trie that match any sub-sequence s of the linearized sequence of T nodes. Whenever we encounter an EOS node e, we verify if any of the structures stored at e matches the sub-tree generated by s. If so, then we have a positive match. As a sentence might embed many shorter utterances, each input T will generally yield multiple matches. For example, querying the tree-trie in Figure 4(c) with the input tree shown in (a) would yield two results, as both *The cat sleeps* and *The cat sleeps under the table* are complete utterances stored in the trie.

Algorithm 3 LOOKUP(T, I): Lookup for matches of subset of tree T in the trie index I.

```
1:  /* Entry point */
2:  L ← T.LINEARIZE()
3:  M ← ∅
4:  LOOKUPRECURSIVE(T, L, 0, I.ROOT(), ∅, M)
5:  return M
6:  /* Procedures */
7:  procedure LOOKUPRECURSIVE(T, L, o, n, P, M)
8:    for all i ∈ [o, L.LENGTH()) do
9:      if n.HASCHILD(L.TOKEN(i)) then
10:       n' ← n.GETCHILD(L.TOKEN(i))
11:       P' ← P ∪ {i}
12:       for all S ∈ n'.TREESTRUCTURES() do
13:         if L.ISCOMPATIBLE(S, P') then
14:           M ← M ∪ {T.GETNODES(P')}
15:       LOOKUPRECURSIVE(L, i, o + 1, n', P', M)
```

Algorithm 3 describes the lookup process in more detail. The first step consists in the linearization of the input tree T. Then, we recursively traverse the trie calling LOOKUPRECURSIVE. The inputs of this procedure are: the input tree T, its linearization L and an offset o (starting at 0), the trie node currently being traversed n (starting with the root), the set of offsets in L that constitute a partial match P (initially empty) and the set of complete matches found M. We recursively traverse all the nodes in the trie that yield a partial match with any sub-sequence of the linearized tokens of T. At each step, we scan all the tokens in L in the range $[o, L.\text{LENGTH}())$ looking for tokens matching any of the children of n. If a matching node is found, a new partial match P' is constructed by extending P with the matching token

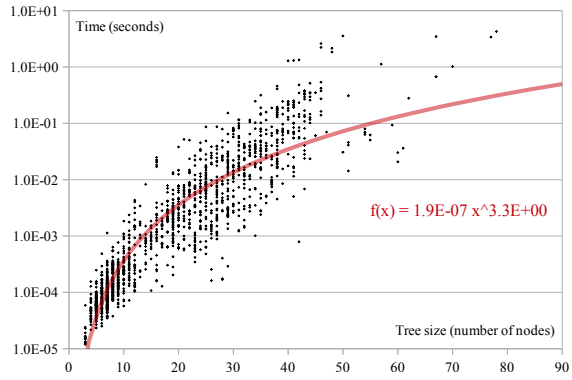


Figure 5: Time complexity of lookup operations for inputs of different sizes.

offset i (line 11), and the recursion continues from the matching trie node n' and offset i (line 15). Every time a partial match is found, we verify if the partial match is compatible with any of the tree structures stored in the matching node. If that is the case, we identify the corresponding set of matching nodes in T and add it to the result M (lines 12-14). A pattern is generated from each complete match returned by LOOKUP after applying a simple heuristic: for each verb node v in the match, we enforce that negations and auxiliaries in T depending from x are also included in the pattern.

Time complexity of lookups. Let k be the maximum fan-out of trie nodes, d be the depth of the trie and n be the size of an input tree (number of nodes). If trie node children are hashed (which has a negative effect on space complexity), then worst case complexity of LOOKUP() is $O(nk)^{d-1}$. If they are stored as sorted lists, as in our memory-efficient implementation, theoretical complexity becomes $O(nk\log(k))^{d-1}$. It should be noted that worst case complexity can only be observed under extremely unlikely circumstances, i.e., that at every step of the recursion all the nodes in the tail of the linearized tree match a child of the current node. Also, in the actual trie used in our experiments the average branching factor k is very small. We observed that a trie storing 400M sentences (170M nodes) has an average branching factor of 1.02. While the root of the trie has unsurprisingly many children (210K, all the observed first sentence words), already at depth 2 the average fan-out is 13.7, and at level 3 it is 4.9.

For an empirical analysis of lookup complexity, Figure 5 plots, in black, wall-clock lookup time as a function of tree size n for a random sample of 1,600 inputs. As shown by the polynomial regression curve (red), observed lookup complexity is approximately cubic with a very small constant factor. In general, we can see that for sentences of common length (20-50 words) a lookup operation can be completed in well under one second.

5 Evaluation

5.1 Experimental settings

All the models for the experiments that we present have been trained using the same corpus of news crawled from the web between 2008 and 2013. The news have been processed with a tokenizer, a sentence splitter (Gillick and Favre, 2009), a part-of-speech tagger and dependency parser (Nivre, 2006), a co-reference resolution module (Haghighi and Klein, 2009) and an entity linker based on Wikipedia and Freebase (Milne and Witten, 2008). We use Freebase types as fine-grained named entity types, so we are also able to label e.g. instances of *sports teams* as such instead of the coarser label ORG.

Next, the news have been grouped based on temporal closeness (Zhang and Weld, 2013) and cosine similarity (using tf·idf weights). For each of the three pattern extraction methods we used the same summarization pipeline (as shown above in Figure 2):

1. Run pattern extraction on the news.
2. For every news collection $Coll$ and entity set E, generate a set containing all the extracted patterns from news in $Coll$ mentioning all the entities in E. These are patterns that are likely to be paraphrasing each other.
3. Run a clustering algorithm to group together patterns that typically co-occur in the sets generated in the previous step. There are many choices for clustering algorithms (Alfonseca et al., 2013; Zhang and Weld, 2013). Following Alfonseca et al. (2013) we use in this work a Noisy-OR Bayesian Network because it has already been applied for abstractive summarization (albeit multi-document), it provides an easily interpretable probabilistic clustering, and training can be easily parallelized to be able to handle large training sets. The hidden events in the Bayesian network represent pattern clusters. When training is done, for each extraction pattern p_j

Original sentence	*Abstractive summary (method)*
Two-time defending overall World Cup champion Marcel Hirscher won the challenging giant slalom on the Gran Risa course with two solid runs Sunday and attributed his victory to a fixed screw in his equipment setup.	Marcel Hirscher has won the giant slalom. (**C**)
Zodiac Aerospace posted a 7.9 percent rise in first-quarter revenue, below market expectations, but reaffirmed its full-year financial targets.	Zodiac Aerospace has reported a rise in profits. (**C**)
Australian free-agent closer Grant Balfour has agreed to terms with the Baltimore Orioles on a two-year deal, the Baltimore Sun reported on Tuesday citing multiple industry sources.	Balfour will join the Baltimore Orioles. (**H**)
Paul Rudd is 'Ant-Man': 5 reasons he needs an 'Agents of SHIELD' appearance.	Paul Rudd to play Ant-Man. (**H**)
Millwall defender Karleigh Osborne has joined Bristol City on a two-and-a-half year deal after a successful loan spell.	Bristol City have signed Karleigh Osborne. (**M**)
Simon Hoggart, one of the Spectator's best-loved columnists, died yesterday after fighting pancreatic cancer for over three years.	Simon Hoggart passed away yesterday. (**M**)

Table 1: Abstraction examples from compression (**C**), heuristic (**H**) and memory-based (**M**) patterns.

Method	*Extractions*	*Abstractions*
HEURISTIC	24,630	956
COMPRESSION	15,687	657
MEMORY-BASED	11,459	967

Table 2: Patterns extracted in each method, before Noisy-OR inference.

and pattern cluster c_i, the network provides $p(p_j|c_i)$ —the probability that c_i will generate p_j— and $p(c_i|p_j)$ —the probability that, given a pattern p_j, c_i was the hidden event that generated it.

At generation time we proceed in the following way:

1. Given the title or first sentence of a news article, run the same pattern extraction method that was used in training and, if possible, obtain a pattern p involving some entities.

2. Find the model clusters that contain this pattern, $C_p = \{c_i \text{ such that } p(c_i|p) > 0\}$.

3. Return a ranked list of model patterns $output = \{(p_j, score(p_j))\}$, scored as follows:

$$score(p_j) = \prod_{c_i \in C_p} p(p_j|c_i)p(c_i|p)$$

where p was the input pattern.

4. Replace the entity placeholders in the top-scored patterns p_j with the entities that were actually mentioned in the input news article.

In all cases the parameters of the network were predefined as 20,000 nodes in the hidden layer (model clusters) and 40 Expectation Maximization (EM) training iterations. Training was distributed across 20 machines with 10 GB of memory each.

For testing we used 37,584 news crawled during December 2013, which had not been used for training the models. Table 3 shows one pattern cluster example from each of the three trained models. The table shows only the surface form of the pattern for simplicity.

Pattern cluster (MEMORY-BASED)
organization$_1$ gets *organization*$_0$ nod for drug
organization$_1$ gets *organization*$_0$ nod for tablets
organization$_0$ approves *organization*$_1$ drug
organizations$_0$ approves *organization*$_1$'s drug
organization$_1$ gets *organization*$_0$ nod for capsules
Pattern cluster (HEURISTIC)
organization$_0$ to buy *organization*$_1$
organization$_0$ to acquire *organization*$_1$
organization$_0$ buys *organization*$_1$
organization$_0$ acquires *organization*$_1$
organization$_0$ to acquire *organizations*$_1$
organization$_0$ buys *organizations*$_1$
organization$_0$ acquires *organizations*$_1$
organization$_0$ agrees to buy *organization*$_1$
organization$_0$ snaps up *organization*$_1$
organization$_0$ to purchase *organizations*$_1$
organization$_0$ is to acquire *organization*$_1$
organization$_0$ has agreed to buy *organization*$_1$
organization$_0$ announces acquisition of *organizations*$_1$
organization$_0$ may bid for *organization*$_1$
organization$_1$ sold to *organization*$_0$
organization$_1$ acquired by *organization*$_0$
Pattern cluster (COMPRESSION)
the *sports team*$_1$ have acquired *person*$_0$ from the *sports team*$_2$
the *sports team*$_1$ acquired *person*$_0$ from the *sports team*$_2$
the *sports team*$_2$ have traded *person*$_0$ to the *sports team*$_1$
sports team$_1$ acquired the rights to *person*$_0$ from *sports team*$_2$
sports team$_2$ acquired from *sports team*$_1$ in exchange for *person*$_0$
sports team$_2$ have acquired from the *sports team*$_1$ in exchange for *person*$_0$

Table 3: Examples of pattern clusters. In each cluster c_i, patterns are sorted by $p(p_j|c_i)$.

5.2 Results

Table 2 shows the number of extracted patterns from the test set, and the number of abstractive event descriptions produced.

As expected, the number of extracted patterns using the memory-based model is smaller than with the two other models, which are based on generic rules and are less restricted in what they can generate. As mentioned, the memory-based model can only extract previously-seen structures. Compared to this model, with heuristics we can obtain patterns for more than twice more news articles. At the same time, looking at the number of summary sentences generated they are comparable, meaning that a larger proportion of the memory-based patterns actually appeared in the pattern clusters and could be used to produce summaries. This is also consistent with the fact that using heuristics the space of extracted patterns is basically unbounded and many new patterns can be generated that were previously unseen –and these cannot generate abstractions. A positive outcome is that restricting the syntactic structure of the extracted patterns to what has been observed in past news does not negatively affect end-to-end coverage when generating the abstractive summaries.

Table 1 shows some of the abstractive summaries generated with the different methods. For manually evaluating their quality, a random sample of 100 original sentences was selected for each method. The top ranked summary for each original sentence was sent to human raters for evaluation, and received three different ratings. None of the raters had any involvement in the development of the work or the writing of the paper, and a constraint was added that no rater could rate more than 50 abstractions. Raters were presented with the original sentence and the compressed abstraction, and were asked to rate it along two dimensions, in both cases using a 5-point Likert scale:

- **Readability**: whether the abstracted compression is grammatically correct.
- **Informativeness**: whether the abstracted compression conveys the most important information from the original sentence.

Inter-judge agreement was measured using the Intra-Class Correlation (ICC) (Shrout and Fleiss, 1979; Cicchetti, 1994). The ICC for readability was 0.37 (95% confidence interval [0.32, 0.41]), and for informativeness it was 0.64 (95% confidence interval [0,60, 0.67]), representing fair and substantial reliability.

Method	*Readability*	*Informativeness*
HEURISTIC	3.95	3.07
COMPRESSION	3.98	2.35
MEMORY-BASED	**4.20**	**3.70**

Table 4: Results for the three methods when rating the top-ranked abstraction.

Table 4 shows the results when rating the top ranked abstraction using either of the three different models for pattern extraction. The abstractions produced with the memory-based method are more readable than those produced with the other two methods (statistically significant with 95% confidence).

Regarding informativeness, the differences between the methods are bigger, because the first two methods have a proportionally larger number of items with a high readability but a low informativeness score. For each method, we have manually reviewed the 25 items where the difference between readability and informativeness was the largest, to understand in which cases grammatical, yet irrelevant compressions are produced. The results are shown in Table 5. *Be+adjective* includes examples where the pattern is of the form *Entity is Adjective*, which the compression-based systems extracts often represents an incomplete extraction. *Wrong inference* contains the cases where patterns that are related but not equivalent are clustered, e.g. *Person arrived in Country* and *Person arrived in Country for talks*. *Info. missing* represents cases where very relevant information has been dropped and the summary sentence is not complete. *Possibility* contains cases where the original sentence described a possibility and the compression states it as a fact, or vice versa. *Disambiguation* are entity disambiguations errors, and *Opposite* contains cases of patterns clustered together that are opposite along some dimension, e.g. *Person quits TV_Program* and *Person to return to TV_Program*.

The method with the largest drop between the readability and informativeness scores is COMPRESSION. As can be seen, many of these mistakes are due to relevant information being missing in the summary sentence. This is also the largest source of errors for the HEURISTIC system. For the MEMORY-BASED system, the drop in read-

Method	*Be+adjective*	*Wrong inference*	*Info. missing*	*Possibility*	*Disambiguation*	*Opposite*
HEURISTIC	0	7	14	3	1	0
COMPRESSION	3	10	10	0	0	2
MEMORY-BASED	0	17	4	2	0	2

Table 5: Sources of errors for the top 25 items with high readability and low informativeness.

Original sentence	*Pattern extracted (method)*	*Abstraction*
David Moyes is happy to use tough love on Adnan Januzaj to ensure the Manchester United youngster fulfils his massive potential.	David Moyes is happy. (**C**)	Fortune will start to favour David Moyes.
The Democratic People's Republic of Korea will "achieve nothing by making threats or provocation," the United States said Friday.	The United States said Friday. (**C, H**)	United States officials said Friday.
EU targets Real and Barca over illegal state aid.	EU targets Real Madrid. (**H**)	EU is going after Real Madrid.
EU warns Israel over settlement construction	EU warns Israel. (**M**)	EU says Israel needs reforms.

Table 6: Examples of compression (**C**), heuristic (**H**) and memory-based (**M**) patterns that led to abstractions with high readability but a low informativeness score. Both incomplete summary sentences and wrong inferences can be observed.

ability score is much smaller, so there were less of these examples. And most of these examples belong to the class of wrong inferences (patterns that are related but not equivalent, so we should not abstract one of them from the other, but they were clustered together in the model). Our conclusion is that the examples with missing information are not such a big problem with the MEMORY-BASED system, as using the trie is an additional safeguard that the generated titles are complete statements, but the method is not preventing the wrong inference errors so this class of errors become the dominant class by a large margin.

Some examples with high readability but low informativeness are shown in Table 6.

6 Conclusions

Most Open-IE systems are based on linguistically-motivated heuristics for learning patterns that express relations between entities or events. However, it is common for these patterns to be incomplete or ungrammatical, and therefore they are not suitable for abstractive summary generation of the relation or event mentioned in the text.

In this paper, we describe a memory-based approach in which we use a corpus of past news to learn valid syntactic sentence structures. We discuss the theoretical time complexity of looking up extraction patterns in a large corpus of syntactic structures stored as a trie and demonstrate empirically that this method is effective in practice. Finally, the evaluation shows that summary sentences produced by this method outperform heuristics and compression-based ones both in terms of readability and informativeness. The problem of generating incomplete summary sentences, which was the main source of informativeness errors for the alternative methods, becomes a minor problem with the memory-based approach. Yet, there are some cases in which also the memory based approach extracts correct but misleading utterances, e.g., a pattern like *PER passed away* from the sentence *PER passed the ball away*. To solve this class of problems, a possible research direction would be the inclusion of more complex linguistic features in the tree-trie, such as verb subcategorization frames.

As another direction for future work, more effort is needed in making sure that no incorrect inferences are made with this model. These happen when a more specific pattern is clustered together with a less specific pattern, or when two non-equivalent patterns often co-occur in news as two events are somewhat correlated in real life, but it is generally incorrect to infer one from the other. Improvements in the pattern-clustering model, outside the scope of this paper, will be required.

References

Enrique Alfonseca, Daniele Pighin, and Guillermo Garrido. 2013. HEADY: News headline abstraction through event pattern clustering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics,* Sofia, Bulgaria, 4–9 August 2013, pages 1243–1253.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics,* Portland, OR, 19–24 June 2011.

Domenic V Cicchetti. 1994. Guidelines, criteria, and rules of thumb for evaluating normed and standardized assessment instruments in psychology. *Psychological Assessment*, 6(4):284.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:399–429.

Hal Daumé III and Daniel Marcu. 2004. A tree-position kernel for document compression. In *Proceedings of the 2004 Document Understanding Conference held at the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics,,* Boston, Mass., 6–7 May 2004.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation,* Genoa, Italy, 22–28 May 2006, pages 449–454.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing,* Edinburgh, UK, 27–29 July 2011, pages 1535–1545.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing,* Seattle, WA, USA, 18–21 October 2013, pages 1481–1491.

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proceedings of the ILP for NLP Workshop,* Boulder, CO, June 4 2009, pages 10–18.

Gregory Grefenstette. 1998. Producing intelligent telegraphic text reduction to provide an audio scanning service for the blind. In *Working Notes of the Workshop on Intelligent Text Summarization,* Palo Alto, Cal., 23 March 1998, pages 111–117.

Aria Haghighi and Dan Klein. 2009. Simple coreference resolution with rich syntactic and semantic features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing,* Singapore, 6-7 August 2009, pages 1152–1161.

Inderjeet Mani. 2001. *Automatic Summarization*. John Benjamins, Amsterdam, Philadelphia.

Ryan McDonald. 2006. Discriminative sentence compression with soft syntactic evidence. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics,* Trento, Italy, 3–7 April 2006, pages 297–304.

Yashar Mehdad, Giuseppe Carenini, and Frank W. Tompa. 2013. Abstractive meeting summarization with entailment and fusion. In *Proceedings of the 14th European Workshop on Natural Language Generation,* Sofia, Bulgaria, 8–9 August, 2013, pages 136–146.

David Milne and Ian H. Witten. 2008. An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In *Proceedings of the AAAI 2008 Workshop on Wikipedia and Artificial Intelligence,* Chicago, IL, 13-14 July, 2008.

Joakim Nivre. 2006. *Inductive Dependency Parsing*. Springer.

Michael Schmitz, Robert Bart, Stephen Soderland, Oren Etzioni, et al. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing,* Jeju Island, Korea, 12–14 July 2012, pages 523–534.

Patrick E Shrout and Joseph L Fleiss. 1979. Intraclass correlations: uses in assessing rater reliability. *Psychological bulletin*, 86(2):420.

Karen Spärck-Jones. 2007. Automatic summarising: A review and discussion of the state of the art. Technical Report UCAM-CL-TR-679, University of Cambridge, Computer Laboratory, Cambridge, U.K.

Ivan Titov and Alexandre Klementiev. 2011. A Bayesian model for unsupervised semantic parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics,* Portland, OR, 19–24 June 2011, pages 1445–1455.

Lu Wang, Hema Raghavan, Vittorio Castelli, Radu Florian, and Claire Cardie. 2013. A sentence compression based framework to query-focused multi-document summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics,* Sofia, Bulgaria, 4–9 August 2013, pages 1384–1394.

Congle Zhang and Daniel S. Weld. 2013. Harvesting parallel news streams to generate paraphrases of event relations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing,* Seattle, WA, USA, 18–21 October 2013, pages 1776–1786.

Hierarchical Summarization: Scaling Up Multi-Document Summarization

Janara Christensen **Stephen Soderland**
Computer Science & Engineering
University of Washington
Seattle, USA
janara@cs.washington.edu
soderlan@cs.washington.edu

Gagan Bansal **Mausam**
Computer Science & Engineering
Indian Institute of Technology
Delhi, India
gaganbansal1993@gmail.com
mausam@cse.iitd.ac.in

Abstract

Multi-document summarization (MDS) systems have been designed for short, unstructured summaries of 10-15 documents, and are inadequate for larger document collections. We propose a new approach to scaling up summarization called *hierarchical summarization*, and present the first implemented system, SUMMA.

SUMMA produces a hierarchy of relatively short summaries, in which the top level provides a general overview and users can navigate the hierarchy to drill down for more details on topics of interest. SUMMA optimizes for coherence as well as coverage of salient information. In an Amazon Mechanical Turk evaluation, users preferred SUMMA ten times as often as flat MDS and three times as often as timelines.

1 Introduction

The explosion in the number of documents on the Web necessitates automated approaches that organize and summarize large document collections on a complex topic. Existing methods for multi-document summarization (MDS) are designed to produce short summaries of 10-15 documents.[1] MDS systems do not scale to data sets ten times larger and proportionately longer summaries: they either cannot run on large input or produce a disorganized summary that is difficult to understand.

We present a novel MDS paradigm, *hierarchical summarization*, which operates on large document collections, creating summaries that organize the information coherently. It mimics how someone with a general interest in a complex topic would learn about it from an expert – first, the expert would provide an overview, and then more specific information about various aspects. Hierarchical summarization has the following novel characteristics:

- The summary is hierarchically organized along one or more organizational principles such as time, location, entities, or events.
- Each non-leaf summary is associated with a set of child summaries where each gives details of an element (e.g. sentence) in the parent summary.
- A user can navigate within the hierarchical summary by clicking on an element of a parent summary to view the associated child summary.

For example, given the topic, "1998 embassy bombings," the first summary (Figure 1) might

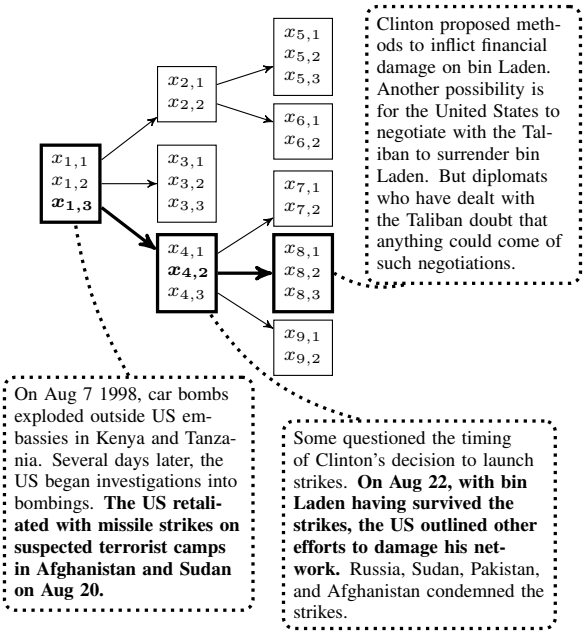


Figure 1: A hierarchical summary of the 1998 embassy bombings. Each rectangle represents a summary and each $x_{i,j}$ is a sentence within a summary. The root summary provides an overview of the events of August 1998. When the third sentence is selected, a more detailed summary of the missile strikes is displayed. Selecting the second sentence of that summary produces a more detailed summary of the US' options.

[1] In the DUC evaluations, summaries have a budget of 665 bytes and cover 10 documents.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 902–912,
Baltimore, Maryland, USA, June 23-25 2014.

mention that the US retaliated by striking Afghanistan and Sudan. The user can click on this information to learn more about these attacks. In this way, the system can present large amounts of information without overwhelming the user, and the user can tailor the output to their interests.

In this paper, we describe SUMMA, the first hierarchical summarization system for multi-document summarization.[2] It operates on a corpus of related news articles. SUMMA hierarchically clusters the sentences by time, and then summarizes the clusters using an objective function that optimizes salience and coherence.

We conducted an Amazon Mechanical Turk (AMT) evaluation where AMT workers compared the output of SUMMA to that of timelines and flat summaries. SUMMA output was judged superior more than three times as often as timelines, and users learned more in twice as many cases. Users overwhelmingly preferred hierarchical summaries to flat summaries (92%) and learned just as much.

Our main contributions are as follows:

- We introduce and formalize the novel task of hierarchical summarization.
- We present SUMMA, the first hierarchical summarization system, which operates on news corpora and summarizes over an order of magnitude more documents than traditional MDS systems, producing summaries an order of magnitude larger.
- We present a user study which demonstrates the value of hierarchical summarization over timelines and flat multi-document summaries in learning about a complex topic.

In the next section, we formalize hierarchical summarization. We then describe our methodology to implement the SUMMA hierarchical summarization system: hierarchical clustering in Section 3 and creating summaries based on that clustering in Section 4. We discuss our experiments in Section 5, related work in Section 6, and conclusions in Section 7.

2 Hierarchical Summarization

We propose a new task for large-scale summarization called *hierarchical summarization*. Input to a hierarchical summarization system is a set of related documents D and a budget b for each summary within the hierarchy (in bytes, words, or sentences). The output is the hierarchical summary H, which we define formally as follows.

[2] http://knowitall.cs.washington.edu/summa/

Definition A *hierarchical summary* H of a document collection D is a set of summaries X organized into a hierarchy. The top of the hierarchy is a summary X_1 representing all of D, and each summary X_i consists of summary units $x_{i,j}$ (*e.g.* the jth sentence of summary i) that point to a child summary, except at the leaf nodes of the hierarchy.

A child summary adds more detail to the information in its parent summary unit. The child summary may include sub-events or background and reactions to the event or topic in the parent.

We define several metrics in Section 4 for a well-constructed hierarchical summary. Each summary should maximize coverage of *salient* information; it should minimize *redundancy*; and it should have *intra-cluster coherence* as well as *parent-to-child coherence*.

Hierarchical summarization has two important strengths in the context of large-scale summarization. First, the information presented at the start is small and grows only as the user directs it, so as not to overwhelm the user. Second, each user directs his or her own experience, so a user interested in one aspect need only explore that section of the data without having to view or understand the entire summary. The parent-to-child links provide a means for a user to navigate, drilling down for more details on topics of interest.

There are several possible organizing principles for the hierarchy – by date, by entities, by locations, or by events. Some organizing principles will fit the data in a document collection better than others. A system may select different organization for different portions of the hierarchy, for example, organizing first by location or prominent entity and then by date for the next level.

3 Hierarchical Clustering

Having defined the task, we now describe the methodology behind our implementation, SUMMA. In future work we intend to design a system that dynamically selects the best organizing principle for each level of the hierarchy. In this first implementation, we have opted for temporal organization, since this is generally the most appropriate for news events.

The problem of hierarchical summarization as described in Section 2 has all of the requirements of MDS, and additional complexities of inducing a hierarchical structure, processing an order of magnitude bigger input, generating a much larger output, and enforcing coherence between parent and

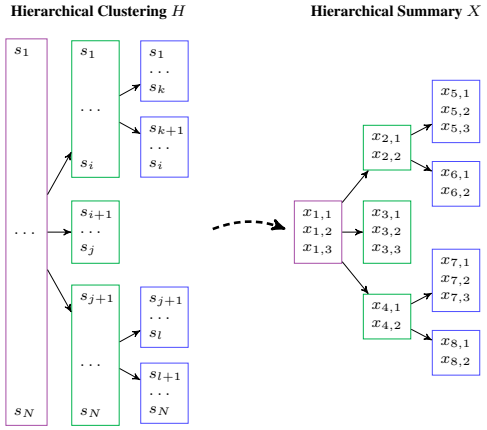


Figure 2: Examples of input and output to hierarchical summarization. The input sentences are $s \in S$, the number of input sentences is N, and the summary sentences are $x \in X$.

child summaries.

We simplify the problem by decomposing it into two steps: hierarchical clustering and summarizing over the clustering (see Figure 2 for an example). A hierarchical clustering is a tree in which if a cluster g_p is the parent of cluster g_c, then each sentence in g_c is also in g_p. This organizes the information into manageable, semantically-related sections and induces a hierarchical structure over the input.

The hierarchical clustering serves as input to the second step – summarizing given the hierarchy. The hierarchical summary follows the hierarchical structure of the clustering. Each node in the hierarchy has an associated flat summary, which summarizes the sentences in that cluster. Moreover, the number of sentences in a flat summary is exactly equal to the number of child clusters of the node, since the user will click a sentence to get to the child summary. See Figure 2 for an illustration of this correspondence.

Because we are interested in *temporal* hierarchical summarization, we hierarchically cluster all the sentences in the input documents by time. Unfortunately, neither agglomerative nor divisive clustering is suitable, since both assume a binary split at each node (Berkhin, 2006). The number of clusters at each split should be what is most natural for the input data. We design a recursive clustering algorithm that automatically chooses the appropriate number of clusters at each split.

Before clustering, we timestamp all sentences. We use SUTime (Chang and Manning, 2012) to normalize temporal references, and we parse the sentences with the Stanford parser (Klein and Manning, 2003) and use a set of simple heuristics to determine if the timestamps in the sentence refer to the root verb. If no timestamp is given, we use the article date.

3.1 Temporal Clustering

After acquiring the timestamps, we must hierarchically cluster the sentences into sets that make sense to summarize together. Since we wish to partition along the temporal dimension, our problem reduces to identifying the best dates at which to split a cluster into subclusters. We identify these dates by looking for bursts of activity.

News tends to be *bursty* – many articles on a topic appear at once and then taper out (Kleinberg, 2002). For example, Figure 3 shows the number of articles per day related to the 1998 embassy bombings published in the New York Times (identified using a key word search). There were two main events – on the 7th, the embassies were bombed and on the 20th, the US retaliated through missile strikes. The figure shows a correspondence between these events and news spikes.

Ideal splits for this example would occur just before each spike in coverage. However, when there is little differentiation in news coverage, we prefer clusters evenly spaced across time. We thus choose clusters $C = \{c_1, \ldots, c_k\}$ as follows:

$$\underset{C}{\text{maximize}} \quad B(C) + \alpha E(C) \tag{1}$$

where C is a clustering, $B(C)$ is the burstiness of the set of clusters, $E(C)$ is the evenness of the clusters, and α is the tradeoff parameter.

$$B(C) = \sum_{c \in C} burst(c) \tag{2}$$

$burst(c)$ is the difference in the number of sentences published the day before the first date in c and the average number of sentences published on the first and second date of c:

$$burst(c) = \frac{pub(d_i) + pub(d_{i+1})}{2} - pub(d_{i-1}) \tag{3}$$

where d is a date indexed over time, such that d_j is a day before d_{j+1}, and d_i is the first date in c. $pub(d_i)$ is the number of sentences published on d_i. The evenness of the split is measured by:

$$E(C) = \min_{c \in C} size(c) \tag{4}$$

where $size(c)$ is the number of dates in cluster c.

We perform hierarchical clustering top-down, at each point solving for Equation 1. α was set using a grid-search over a development set.

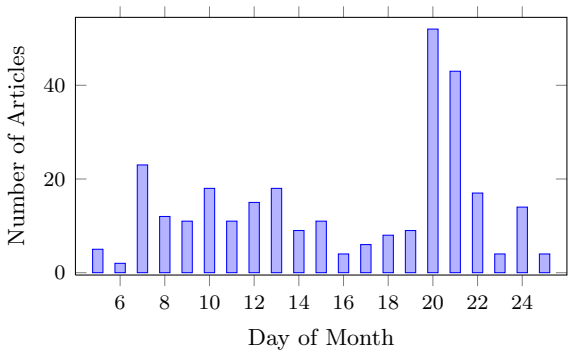


Figure 3: News coverage by date for the embassy bombings in Tanzania and Kenya. There are spikes in the number of articles published at the two major events.

3.2 Choosing the number of clusters

We cannot know *a priori* the number of clusters for a given topic. However, when the number of clusters is too large for the given summary budget, the sentences will have to be too short, and when the number of clusters is too small, we will not use enough of the budget. We set the maximum number of clusters k_{max} and minimum number of clusters k_{min} to be a function of the budget b and the average sentence length in the cluster s_{avg}, such that $k_{max} \cdot s_{avg} \leq b$ and $k_{min} \cdot s_{avg} \geq b/2$.

Given a maximum and minimum number of clusters, we must determine the appropriate number of clusters. At each level, we cluster the sentences by the method described above and choose the number of clusters k according to the gap statistic (Tibshirani et al., 2000). Specifically, for each level, the algorithm will cluster repeatedly with k varying from the minimum to the maximum. The algorithm will return the k that maximizes the gap statistic:

$$Gap_n(k) = E_n^*\{\log(W_k)\} - log(W_k) \qquad (5)$$

where W_k is the score for the clusters computed with Equation 1, and E_n^* is the expectation under a sample of size n from a reference distribution.

Ideally, the maximum depth of the clustering would be a function of the number of sentences in each cluster, but in our implementation, we set the maximum depth to three, which works well for the size of the datasets we use (300 articles).

4 Summarizing within the Hierarchy

After the sentences are clustered, we have a structure for the hierarchical summary that dictates the number of summaries and the number of sentences in each summary. We also have the set of sentences from which each summary is drawn.

Intuitively, each cluster summary in the hierarchical summary should convey the most **salient** information in that cluster. Furthermore, the hierarchical summary should not include **redundant** sentences. A hierarchical summary that is only salient and nonredundant may still not be suitable if the sentences within a cluster summary are disconnected or if the parent sentence for a summary does not relate to the child summary. Thus, a hierarchical summary must also have **intra-cluster coherence** and **parent-to-child coherence**.

4.1 Salience

Salience is the value of each sentence to the topic from which the documents are drawn. We measure salience of a summary ($Sal(X)$) as the sum of the saliences of individual sentences ($\sum_i Sal(x_i)$). Following previous research in MDS, we computed individual saliences using a linear regression classifier trained on ROUGE scores over the DUC'03 dataset (Lin, 2004; Christensen et al., 2013). This method finds those sentences more salient that mention nouns or verbs that occur frequently in the cluster.

In preliminary experiments, we noticed that many sentences that were *reaction* sentences were given a higher salience than *action* sentences. For example, the reaction sentence, "President Clinton vowed to track down the perpetrators behind the bombs that exploded outside the embassies in Tanzania and Kenya on Friday," would have a higher score than the *action* sentence, "Bombs exploded outside the embassies in Tanzania and Kenya on Friday." This problem occurs because the first sentence has a higher ROUGE score (it covers more important words than the second sentence). To adjust for this problem, we use only words identified in the main clause (heuristically identified via the parse tree) to compute our salience scores.

4.2 Redundancy

We identify redundant sentences using a linear regression classifier trained on a manually labeled subset of the DUC'03 sentences. The features include shared noun counts, sentence length, TF*IDF cosine similarity, timestamp difference, and features drawn from information extraction such as number of shared tuples in Open IE (Mausam et al., 2012).

4.3 Summary Coherence

We require two types of coherence: coherence between the parent and child summaries and coherence within each summary X_i.

We rely on the approximate discourse graph (ADG) that was proposed in (Christensen et al., 2013) as the basis for measuring coherence. Each node in the ADG is a sentence from the dataset. An edge from sentence s_i to s_j with positive weight indicates that s_j may follow s_i in a coherent summary, *e.g.* continued mention of an event or entity, or coreference link between s_i and s_j. A negative edge indicates an unfulfilled discourse cue or co-reference mention.

Parent-to-Child Coherence: Users navigate the hierarchical summary from parent sentence to child summary, so if the parent sentence bears no relation to the child summary, the user will be understandably confused. The parent sentence must have positive evidence of coherence with the sentences in its child summary.

We estimate parent to child coherence as the coherence between a parent sentence and each sentence in its child summary as:

$$PCoh(X) = \sum_{c \in C} \sum_{i=1..|X_c|} w_{G+}(x_c^p, x_{c,i})) \quad (6)$$

where x_c^p is the parent sentence for cluster c and $w_{G+}(x_c^p, x_{c,i})$ is the sum of the positive edge weights from x_c^p to $x_{c,i}$ in the ADG G.

Intra-cluster Coherence: In traditional MDS, the documents are usually quite focused, allowing for highly focused summaries. In hierarchical summarization, however, a cluster summary may span hundreds of documents and a wide range of information. For this reason, we may consider a summary acceptable even if it has limited positive evidence of coherence in the ADG, as long as there is no negative evidence in the form of negative edges. For example, the following is a reasonable summary for events spanning two weeks:

s_1 *Bombs exploded at two US embassies.*

s_2 *US missiles struck in Afghanistan and Sudan.*

Our measure of intra-cluster coherence minimizes the number of *missing references*. These are coreference mentions or discourse cues where none of the sentences read before (either in an ancestor summary or in the current summary) contain an antecedent:

$$CCoh(X) = -\sum_{c \in C} \sum_{i=1..|X_c|} \#missingRef(x_{c,i}) \quad (7)$$

4.4 Objective Function

Having estimated salience, redundancy, and two forms of coherence, we can now put this information together into a single objective function that measures the quality of a candidate hierarchical summary.

Intuitively, the objective function should balance salience and coherence. Furthermore, the summary should not contain redundant information and each cluster summary should honor the given budget, i.e., maximum summary length b. We treat redundancy and budget as hard constraints and coherence and salience as soft constraints. Lastly, we require that sentences are drawn from the cluster that they represent and that the number of sentences in the summary corresponding to each non-leaf cluster c is equivalent to the number of child clusters of c. We optimize:

$$\begin{aligned} \textbf{maximize:} \quad & F(x) \triangleq Sal(X) + \beta PCoh(X) + \gamma CCoh(X) \\ s.t. \quad & \forall c \in C : \textstyle\sum_{i=1..|X_c|} len(x_{c,i}) < b \\ & \forall x_i, x_j \in X : \text{redundant}(x_i, x_j) = 0 \\ & \forall c \in C, \forall x_c \in X_c : x_c \in c \\ & \forall c \in C : |X_c| = \#children(c) \end{aligned}$$

The tradeoff parameters β and γ were set based on a development set.

4.5 Algorithm

Optimizing this objective function is NP-hard, so we approximate a solution by using beam search over the space of partial hierarchical summaries. Notice the contribution from a sentence depends on individual salience, coherence ($CCoh$) based on sentences visible on the user path down the hierarchy to this sentence, and coherence ($PCoh$) based on its parent sentence and its child summary. Since most of the sentence contributions depend on the path from the root to the sentence, we build our partial summary by incrementally adding a sentence top-down in the hierarchy and from first sentence to last within a cluster summary.

To account for $PCoh$, we estimate the contribution of the sentence by jointly identifying its best child summary. However, we do not fix the child summary at this time – we simply use it to estimate $PCoh$ when using that sentence. Since computing the best child summary is also intractable we approximate a solution by a local search algorithm over the child cluster.

Overall, our algorithm is a two level nested search algorithm – beam search in the outer loop to

search through the space of partial summaries and local search (hill climbing with random restarts) in the inner loop to pick the best sentence to add to the existing partial summary. We use a beam of size ten in our implementation.

5 Experiments

Our experiments are designed to evaluate how effective hierarchical summarization is in summarizing a large, complex topic and how well this helps users learn about the topic. Our evaluation addresses the following questions:

- Do users prefer hierarchical summaries for topic exploration? (Section 5.1)
- Are hierarchical summaries more effective than other methods for learning about complex events? (Section 5.2)
- How informative are the hierarchical summaries compared to the other methods? (Section 5.3)
- How coherent is the hierarchical structure in the summaries? (Section 5.4)

We compared SUMMA against two baseline systems which represent the main NLP methods for large-scale summarization: an algorithm for creating timelines over sentences (Chieu and Lee, 2004),[3] and a state-of-the-art flat MDS system (Lin and Bilmes, 2011).[4] Each system was given the same budget (over 10 times the traditional MDS budget, which is 665 bytes).

We evaluated the questions on ten news topics, representing a range of tasks: (1) Pope John Paul II's death and the 2005 Papal Conclave, (2) Bush v. Gore, (3) the Tulip Revolution, (4) Daniel Pearl's kidnapping, (5) the Lockerbie bombing handover of suspects, (6) the Kargil War, (7) NATO's bombing of Yugoslavia in 1999, (8) Pinochet's arrest in London, (9) the 2005 London bombings, and (10) the crash and investigation of SwissAir Flight 111. We chose topics containing a set of related events that unfolded over several months and were prominent enough to be reported in at least 300 articles.

We drew our articles from the Gigaword corpus, which contains articles from the New York Times and other major newspapers. For each topic, we used the 300 documents that best matched a key word search. We selected topics which were between five and fifteen years old so that evaluators would have relatively less pre-existing knowledge about the topic.

5.1 User Preference

In our first experiment, we simply wished to evaluate which system users most prefer. We hired Amazon Mechanical Turk (AMT) workers and assigned two topics to each worker. We paired up workers such that one worker would see output from SUMMA for the first topic and a competing system for the second and the other worker would see the reverse. For quality control, we asked workers to complete a qualification task first, in which they were required to write a short summary of a news article. We also manually removed spam from our results. Previous work has used AMT workers for summary evaluations and has shown high correlations with expert ratings (Christensen et al., 2013). Five workers were hired to view each topic-system pair.

We asked the workers to choose which format they preferred and to explain why. The results are as follows:

SUMMA	**76%**	TIMELINE	24%
SUMMA	**92%**	FLAT-MDS	8%

Users preferred the hierarchical summaries three times more often than timelines and over ten times more often than flat summaries. When we examined the reasons given by the users, we found that the people who preferred the hierarchical summaries liked that they gave a big picture overview and were then allowed to drill down deeper. Some also explained that it was easier to remember information when presented with the overview first. Typical responses included, "Could gather and absorb the information at my own pace," and, "Easier to follow and understand." When users preferred the timelines, they usually remarked that it was more familiar, i.e. "I liked the familiarity of the format. I am used to these timelines and they feel comfortable." Users complained that the flat summaries were disjointed, confusing, and very frustrating to read.

5.2 Knowledge Acquisition

Evaluating how much a user learned is inherently difficult, more so when the goal is to allow the user the freedom to explore information based on individual interest. For this reason, instead of asking a set of predefined questions, we assess the knowl-

[3] Unfortunately, we were unable to obtain more recent timeline systems from authors of the systems.

[4] (Christensen et al., 2013) is a state-of-the-art coherent MDS system, but does not scale to 300 documents.

edge gain by following the methodology of (Shahaf et al., 2012) – asking users to write a paragraph summarizing the information learned.

Using the same setup as in the previous experiment, for each topic, five AMT workers spent three minutes reading through a timeline or summary and were then asked to write a description of what they had learned. Workers were not allowed to see the timeline or summary while writing. We collected five descriptions for each topic-system combination. We then asked other AMT workers to read and compare the descriptions written by the first set of workers. Each evaluator was presented with a corresponding Wikipedia article and descriptions from a pair of users (timeline vs. SUMMA or flat MDS vs. SUMMA). The descriptions were randomly ordered to remove bias. The workers were asked which user appeared to have learned more and why. For each pair of descriptions, four workers evaluated the pair. Standard checks such as approval rating, location filtering, etc. were used for removing spam. The results of this experiment are as follows:

Prefer		Indiff.	Prefer	
SUMMA	**58%**	17%	TIMELINE	25%
SUMMA	**40%**	22%	FLAT-MDS	38%

Descriptions written by workers using SUMMA were preferred over twice as often as those from timelines. We looked more closely at those cases where the participants either preferred the timelines or were indifferent and found that this preference was most common when the topic was not dominated by a few major events, but was instead a series of similarly important events. For example, in the kidnapping and beheading of Daniel Pearl there were two or three obviously major events, whereas in the Kargil War there were many smaller important events. In latter cases, the hierarchical summaries provided little advantage over the timelines because it was more difficult to arrange the sentences hierarchically.

Since SUMMA was judged to be so much superior to flat MDS systems in Section 5.1, it is surprising that users descriptions from flat MDS were preferred nearly as often as those from SUMMA. While the flat summaries were disjointed, they were good at including salient information, with the most salient tending to be near the start of the summary. Thus, descriptions from both SUMMA and flat MDS generally covered the most salient information.

5.3 Informativeness

In this experiment, we assess the salience of the information captured by the different systems, and the ability of SUMMA to organize the information so that more important information is placed at higher levels.

ROUGE Evaluation: We first automatically assessed informativeness by calculating the ROUGE-1 scores of the output of each of the systems. For the gold standard comparison summary, we use the Wikipedia articles for the topics.[5] Note that there is no good translation of ROUGE for hierarchical summarization. Thus, we simply use the traditional ROUGE metric, which will not capture any of the hierarchical format. This score will essentially serve as a rough measure of coverage of the entire summary to the Wikipedia article. The scores for each of the systems are as follows:

	P	R	F1
SUMMA	0.25	**0.67**	0.31
TIMELINE	0.28	0.65	0.33
FLAT-MDS	**0.30**	0.64	**0.34**

None of the differences are significant. From this evaluation, one can gather that the systems have similar coverage of the Wikipedia articles.

Manual Evaluation: While ROUGE serves as a rough measure of coverage, we were interested in gathering more fine-grained information on the informativeness of each system. We performed an additional manual evaluation that assesses the recall of important events for each system.

We first identified which events were most important in a news story. Because reading 300 articles per topic is impractical, we asked AMT workers to read a Wikipedia article on the same topic and then identify the three most important events and the five most important secondary events. We aggregated responses from ten workers per topic and chose the three most common primary and five most common secondary events.

One of the authors then manually identified the presence of these events in the hierarchical summaries, the timelines and the flat MDS summaries. Below we show event recall (the percentage of the events that were mentioned).

[5] We excluded one topic (the handover of the Lockerbie bombing suspects) because the corresponding Wikipedia article had insufficient information.

Events	SUMMA	TIMELINE	FLAT-MDS
Prim.	**96%**	74%	93%
Sec.	**76%**	53%	64%

The difference in recall between SUMMA and TIMELINE was significant in both cases, and the difference between SUMMA and FLAT-MDS was not. In general, the flat summaries were quite redundant, which contributed to the slightly lower event recall. The timelines, on the other hand, were both incoherent and at the same time reported less important facts.

We also evaluated at what level in the hierarchy the events were identified for the hierarchical summaries. The event recall shows the percentage of events mentioned at that level or above in the hierarchical summary:

Events	Level 1	Level 2	Level 3
Prim.	63%	81%	96%
Sec.	27%	51%	76%

81% of the primary events are present in the first or second level, and 76% of the secondary events are mentioned by the third level. While recognizing primary events is relatively simple because they are repeated frequently, identification of important secondary events often requires external knowledge.

5.4 Parent-to-Child Coherence

We next tested the hierarchical coherence. One of the authors graded how much each non-leaf sentence in a summary was coherent with its child summary on a scale of one to five, with one being incoherent and five being perfectly coherent. We used the coherence scale from DUC'04.[6]

	Level 1	Level 2
Coherence	3.8	3.4

We found that for the top level of the summary, the parent sentence generally represented the most important event in the cluster and the child summary usually expressed details or reactions of the event. The lower coherence scores were often the result of too few lexical connections or lack of a theme or story. While the facts of the sentences made sense together, the summaries sometimes did not read as if they were written by a human, but as a series of disparate sentences.

For the second level, the problems were more basic. The parent sentence occasionally expressed a less important fact that the child summary did not then expand on or, more commonly, the child summary was not focused enough. This result stems from two problems in our algorithm. First, summarizing sentences are rare, making good choices for parent sentences difficult to find. The second problem relates to the difficulty in identifying whether two sentences are on the same topic. For example, suppose the parent sentence is, "A Swissair plane Wednesday night crashed off Nova Scotia, Canada." A very good child sentence is, "The airline confirmed that all passengers died." However, based on their surface features, the sentence, "A plane made an unscheduled landing after a Swissair plane crashed off the coast of Canada," appears to be a better choice.

Even though there is scope for improvement, we find these coherence scores encouraging for a first algorithm for the task.

6 Related Work

Traditional approaches to large-scale summarization have included flat summaries and timelines. There are two primary shortcomings to these approaches: first, they require the user to sort through large amounts of potentially overwhelming information, and second, the output is static – users with different interests will see the same information. Below we describe related work on traditional MDS, structured summaries, timelines, discovering threads of documents and the uses of hierarchies in generating summaries.

6.1 Traditional MDS

Traditionally, MDS systems have focused on three to six sentence summaries covering 10-15 documents. Most extractive summarization research aims to maximize coverage while reducing redundancy (e.g. (Carbonell and Goldstein, 1998; Saggion and Gaizauskas, 2004; Radev et al., 2004)). Lin and Bilmes (2011) proposed a state-of-the-art system that uses submodularity in sentence selection to accomplish these goals. Christensen et al. (2013) presented an algorithm for coherent MDS, but it does not scale to larger output.

Structured Summaries: Some research has explored generating structured summaries. These approaches attempt to identify major aspects of a topic, but do not compile content to describe those aspects. Rather, they rely on pre-existing, labeled paragraphs (for example, a paragraph titled, "Symptoms of Meningitis"). Aspects are identified either by a training corpus of articles in the

[6] http://duc.nist.gov/duc2004/quality.questions.txt

same domain (Sauper and Barzilay, 2009), by an entity-aspect LDA model (Li et al., 2010), or by Wikipedia templates of related topics (Yao et al., 2011). These methods assume a common structure for all topics in a category, and do not allow for more than two levels in the structure.

Timeline Generation: Recent papers in timeline generation have emphasized the relationship with summarization. Yan et al. (2011b) balanced coherence and diversity to create timelines, Yan et al. (2011a) used inter-date and intra-date sentence dependencies, and Chieu and Lee (2004) used sentence similarity. Others have emphasized identifying important dates, primarily by bursts of news (Swan and Allen, 2000; Akcora et al., 2010; Hu et al., 2011; Kessler et al., 2012). While timelines can be useful for understanding events, they do not generalize to other domains. Additionally, long timelines can be overwhelming, short timelines have low information content, and there is no method for personalized exploration.

Document Threads: A related track of research investigates discovering threads of documents. While we aim to summarize collections of information, this track seeks to identify relationships between documents. This research operates on the document level, while ours operates on the sentence level. Shahaf and Guestrin (2010) formalized the characteristics of a good chain of articles and proposed an algorithm to connect two specified articles. Gillenwater et al. (2012) proposed a probabilistic technique for extracting a diverse set of threads from a given collection. Shahaf et al. (2012) extended work on coherent threads to finding coherent maps of documents, where a map is set of intersecting threads representing how the threads interact and relate.

Summarization and Hierarchies: A few papers have examined the relationship between summarization and hierarchies. Some focused on creating a hierarchical summary of a single document (Buyukkokten et al., 2001; Otterbacher et al., 2006), relying on the structure inherent in single documents. Others investigated creating hierarchies of words or phrases to organize documents (Lawrie et al., 2001; Lawrie, 2003; Takahashi et al., 2007; Haghighi and Vanderwende, 2009).

Other research identifies the hierarchical structure of the documents and generates a summary that prioritizes more general information according to the structure (Ouyang et al., 2009; Celikyilmaz and Hakkani-Tur, 2010), or gains coverage by drawing sentences from different parts of the hierarchy (Yang and Wang, 2003; Wang et al., 2006).

7 Conclusions

We have introduced a new paradigm for large-scale summarization called hierarchical summarization, which allows a user to navigate a hierarchy of relatively short summaries. We present SUMMA, an implemented hierarchical news summarization system,[7] and demonstrate its effectiveness in a user study that compares SUMMA with a timeline system and a flat MDS system. When compared to timelines, users learned more with SUMMA in twice as many cases, and SUMMA was preferred more than three times as often. When compared to flat summaries, users overwhelming preferred SUMMA and learned just as much.

This first implementation performs temporal clustering – in future work, we will investigate dynamically selecting an organizing principle that is best suited to the data at each level of the hierarchy: by entity, by location, by event, or by date. We also intend to scale the system to even larger document collections, and explore joint clustering and summarization. Lastly, we plan to research hierarchical summarization in other domains.

Acknowledgments

We thank Amitabha Bagchi, Niranjan Balasubramanian, Danish Contractor, Oren Etzioni, Tony Fader, Carlos Guestrin, Prachi Jain, Lucy Vanderwende, Luke Zettlemoyer, and the anonymous reviewers for their helpful suggestions and feedback. We thank Hui Lin and Jeff Bilmes for providing us with their code. This research was supported in part by ARO contract W911NF-13-1-0246, DARPA Air Force Research Laboratory (AFRL) contract FA8750-13-2-0019, UW-IITD subcontract RP02815, and the Yahoo! Faculty Research and Engagement Award. This paper is also supported in part by the Intelligence Advanced Research Projects Activity (IARPA) via AFRL contract number FA8650-10-C-7058. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, AFRL, or the U.S. Government.

[7] http://knowitall.cs.washington.edu/summa/

References

C. G. Akcora, M. A. Bayir, M. Demirbas, and H. Ferhatosmanoglu. 2010. Identifying breakpoints in public opinion. In *1st KDD Workshop on Social Media Analytics*.

Berkhin Berkhin. 2006. A survey of clustering data mining techniques. *Grouping Multidimensional Data*, pages 25–71.

Orkut Buyukkokten, Hector Garcia-Molina, and Andreas Paepcke. 2001. Seeing the whole in parts: Text summarization for web browsing on handheld devices. In *Proceedings of WWW 2001*, pages 652–662.

Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of SIGIR 1998*, pages 335–336.

Asli Celikyilmaz and Dilek Hakkani-Tur. 2010. A hybrid hierarchical model for multi-document summarization. In *Proceedings of ACL 2010*, pages 815–824.

Angel Chang and Christopher Manning. 2012. SUTime: A library for recognizing and normalizing time expressions. In *Proceedings of LREC 2012*.

Hai Leong Chieu and Yoong Keok Lee. 2004. Query based event extraction along a timeline. In *Proceedings of SIGIR 2004*, pages 425–432.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2013. Towards coherent multi-document summarization. In *Proceedings of NAACL 2013*.

Jennifer Gillenwater, Alex Kulesza, and Ben Taskar. 2012. Discovering diverse and salient threads in document collections. In *Proceedings of EMNLP-CoNLL 2012*, pages 710–720.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. *Proceedings of NAACL 2009*, pages 362–370.

Po Hu, Minlie Huang, Peng Xu, Weichang Li, Adam K. Usadi, and Xiaoyan Zhu. 2011. Generating breakpoint-based timeline overview for news topic retrospection. In *Proceedings of ICDM 2011*.

Remy Kessler, Xavier Tannier, Caroline Hagège, Véronique Moriceau, and André Bittar. 2012. Finding salient dates for building thematic timelines. In *Proceedings of ACL 2012*, pages 730–739.

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

Jon Kleinberg. 2002. Bursty and hierarchical structure in streams. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 91–101.

Dawn Lawrie, W. Bruce Croft, and Arnold Rosenberg. 2001. Finding topic words for hierarchical summarization. In *Proceedings of SIGIR '01*, pages 349–357.

Dawn J. Lawrie. 2003. *Language models for hierarchical summarization*. Ph.D. thesis, University of Massachusetts Amherst.

Peng Li, Jing Jiang, and Yinglin Wang. 2010. Generating templates of entity summaries with an entity-aspect model and pattern mining. In *Proceedings of ACL 2010*, pages 640–649.

Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *Proceedings of ACL 2011*, pages 510–520.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of EMNLP 2012*, pages 523–534.

Jahna Otterbacher, Dragomir Radev, and Omer Kareem. 2006. News to go: Hierarchical text summarization for mobile devices. In *Proceedings of SIGIR 2006*, pages 589–596.

You Ouyang, Wenji Li, and Qin Lu. 2009. An integrated multi-document summarization approach based on word hierarchical representation. In *Proceedings of the ACLShort 2009*, pages 113–116.

Dragomir R. Radev, Hongyan Jing, Malgorzata Stys, and Daniel Tam. 2004. Centroid-based summarization of multiple documents. *Information Processing and Management*, 40(6):919–938.

Horacio Saggion and Robert Gaizauskas. 2004. Multi-document summarization by cluster/profile relevance and redundancy removal. In *Proceedings of DUC 2004*.

Christina Sauper and Regina Barzilay. 2009. Automatically generating Wikipedia articles: A structure-aware approach. In *Proceedings of ACL 2009*, pages 208–216.

Dafna Shahaf and Carlos Guestrin. 2010. Connecting the dots between news articles. In *Proceedings of KDD 2010*, pages 623–632.

Dafna Shahaf, Carlos Guestrin, and Eric Horvitz. 2012. Trains of thought: Generating information maps. In *Proceedings of WWW 2012*.

Russell Swan and James Allen. 2000. Automatic generation of overview timelines. In *Proceedings of SIGIR 2000*, pages 49–56.

Kou Takahashi, Takao Miura, and Isamu Shioya. 2007. Hierarchical summarizing and evaluating for web pages. In *Proceedings of the 1st workshop on emerging research opportunities for Web Data Management (EROW 2007)*.

Robert Tibshirani, Guenther Walther, and Trevor Hastie. 2000. Estimating the number of clusters in a dataset via the gap statistic. *Journal of the Royal Statistical Society, Series B*, 32(2):411–423.

Fu Lee Wang, Christopher C. Yang, and Xiaodong Shi. 2006. Multi-document summarization for terrorism information extraction. In *Proceedings of ISI'06*.

Rui Yan, Liang Kong, Congrui Huang, Xiaojun Wan, Xiaoming Li, and Yan Zhang. 2011a. Timeline generation through evolutionary trans-temporal summarization. In *Proceedings of EMNLP 2011*, pages 433–443.

Rui Yan, Xiaojun Wan, Jahna Otterbacher, Liang Kong, Xiaoming Li, and Yan Zhang. 2011b. Evolutionary timeline summarization: A balanced optimization framework via iterative substitution. In *Proceeding of SIGIR 2011*, pages 745–754.

Christopher C. Yang and Fu Lee Wang. 2003. Fractal summarization: summarization based on fractal theory. In *Proceedings of SIGIR 2003*, pages 391–392.

Conglei Yao, Xu Jia, Sicong Shou, Shicong Feng, Feng Zhou, and Hongyan Liu. 2011. Autopedia: Automatic domain-independent wikipedia article generation. In *Proceedings of WWW 2011*, pages 161–162.

Query-Chain Focused Summarization

Tal Baumel
Dept. of Computer Science
Ben-Gurion University
Beer-Sheva, Israel
talbau@cs.bgu.ac.il

Raphael Cohen
Dept. of Computer Science
Ben-Gurion University
Beer-Sheva, Israel
cohenrap@cs.bgu.ac.il

Michael Elhadad
Dept. of Computer Science
Ben-Gurion University
Beer-Sheva, Israel
elhadad@cs.bgu.ac.il

Abstract

Update summarization is a form of multi-document summarization where a document set must be summarized in the context of other documents assumed to be known. Efficient update summarization must focus on identifying new information and avoiding repetition of known information. In *Query-focused summarization*, the task is to produce a summary as an answer to a given query. We introduce a new task, *Query-Chain Summarization*, which combines aspects of the two previous tasks: starting from a given document set, increasingly specific queries are considered, and a new summary is produced at each step. This process models *exploratory search*: a user explores a new topic by submitting a sequence of queries, inspecting a summary of the result set and phrasing a new query at each step. We present a novel dataset comprising 22 query-chains sessions of length up to 3 with 3 matching human summaries each in the consumer-health domain. Our analysis demonstrates that summaries produced in the context of such exploratory process are different from informative summaries. We present an algorithm for Query-Chain Summarization based on a new LDA topic model variant. Evaluation indicates the algorithm improves on strong baselines.

1 Introduction

In the past 10 years, the general objective of text summarization has been refined into more specific tasks. Such summarization tasks include: (i) Generic Multi Document Summarization: aims at summarizing a cluster of topically related documents, such as the top results of a search engine query; (ii) in Update Summarization, a set of documents is summarized while assuming the user has already read a summary of earlier documents on the same topic; (iii) in Query-Focused Summarization, the summary of a documents set is produced to convey an informative answer in the context of a specific query. The importance of these specialized tasks is that they help us distinguish criteria that lead to the selection of content in a summary: centrality, novelty, relevance, and techniques to avoid redundancy.

We present in this paper a variant summarization task which combines the two aspects of update and query-focused summarization. The task is related to *exploratory search* (Marchionini, 2006). In contrast to classical information seeking, in exploratory search, the user is uncertain about the information available, and aims at learning and understanding a new topic (White and Roth, 2009). In typical exploratory search behavior, a user posts a series of queries, and based on information gathered at each step, decides how to further explore a set of documents. The metaphor of *berrypicking* introduced in (Bates, 1989) captures this interactive process. At each step, the user may *zoom in* to a more specific information need, *zoom out* to a more general query, or *pan sideways*, in order to investigate a new aspect of the topic.

We define *Query-Chain Focused Summarization* as follows: for each query in an exploratory search session, we aim to extract a summary that answers the information need of the user, in a manner similar to *Query-Focused Summarization*, while not repeating information already provided in previous steps, in a manner similar to *Update Summarization*. In contrast to query-focused summarization, the context of a sum-

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 913–922,
Baltimore, Maryland, USA, June 23-25 2014.

mary is not a single query, but the set of queries that led to the current step, their result sets and the corresponding summaries.

We have constructed a novel dataset of Query-Sets with matching manual summarizations in the consumer health domain (Cline and Haynes, 2001). Queries are extracted from PubMed search logs (Dogan et al., 2009). We have analyzed this manual dataset and confirm that summaries written in the context of berry-picking are markedly different from those written for similar queries on the same document set, but without the query-chain context.

We have adapted well-known multi-document algorithms to the task, and present baseline algorithms based on LexRank (Erkan and Radev, 2004), KLSum and TopicSum (Haghighi and Vanderwende, 2009). We introduce a new algorithm to address the task of Query-Chain Focused Summarization, based on a new LDA topic model variant, and present an evaluation which demonstrates it improves on these baselines.

The paper is structured as follows. Section 2 formulates the task of Query-Chain Focused Summarization. Section 3 reviews related work. In Section 4, we describe the data collection process and the resulting dataset. We then present our algorithm, as well as the baseline algorithms used for evaluation. We conclude with evaluation and discussion.

2 Query- Chain Summarization

In this work, we focus on the *zoom in* aspect of the exploratory search process described above. We formulate the Query-Chain Focused Summarization (QCFS) task as follows:

Given an ordered chain of queries Q and a set of documents D, for each query $q_i \in Q$ a summary S_i is generated from D answering q_i under the assumption that the user has already read the summaries S_{i-1} for queries $q_0 \ldots q_{i-1}$.

A typical example of query chain in the consumer health domain we investigate includes the following 3 successive queries: (*Causes of asthma, Asthma and Allergy*, Asthma *and Mold Allergy*). We consider a single set of documents relevant to the domain of Asthma as the reference set *D*. The QCFS task consists of generating one summary of *D* as an answer to each query, so that the successive answers do not repeat information already provided in a previous answer.

3 Previous Work

We first review the closely related tasks of Update Summarization and Query-Focused Summarization. We also review key summarization algorithms that we have selected as baseline and adapted to the QCFS task.

Update Summarization focuses on identifying new information relative to a previous body of information, modeled as a set of documents. It has been introduced in shared tasks in DUC 2007 and TAC 2008. This task consists of producing a multi-document summary for a document set on a specific topic, and then a multi-document summary for a different set of articles on the same topic published at later dates. This task helps us understand how update summaries identified and focused on new information while reducing redundancy compared to the original summaries.

The TAC 2008 dataset includes 48 sets of 20 documents, each cluster split in two subsets of 10 documents (called A and B). Subset B documents were more recent. Original summaries were generated for the A subsets and update summaries were then produced for the B subsets. Human summaries and candidate systems are evaluated using the Pyramid method (Nenkova and Passonneau, 2004). For automatic evaluation, ROUGE (Lin, 2004) variants have been proposed (Conroy et al., 2011). In contrast to this setup, QCFS distinguishes the subsets of documents considered at each step of the process by facets of the underlying topic, and not by chronology. In addition, the document subsets are not identified as part of the task in QCFS (as opposed to the explicit split in A and B subsets in Update Summarization).

Most systems working on Update Summarization have focused on removing redundancy. DualSum (Delort and Alfonseca, 2012) is notable in attempting to directly model novelty using a specialized topic-model to distinguish words expressing background information and those introducing new information in each document.

In Query-Focused Summarization (QFS), the task consists of identifying information in a document set that is most relevant to a given query.

This differs from generic summarization, where one attempts to identify central information. QFS helps us distinguish models of relevance and centrality. Unfortunately, detailed analysis of the datasets produced for QFS indicates that these two notions are not strongly distinguished in practice: (Gupta et al., 2007) observed that in QFS datasets, up to 57% of the words in the document sets were closely related to the query (through simple query expansion). They note that as a consequence, a generic summarizer forms a strong baseline for such biased QFS tasks.

We address this limitation of existing QFS datasets in our definition of QCFS: we identify a chain of at least 3 related queries which focus on different facets of the same central topic and require the generation of distinct summaries for each query, with little repetition across the steps.

A specific evaluation aspect of QFS measures responsiveness (how well the summary answers the specific query). QFS must rely on Information Retrieval techniques to overcome the scarceness of the query to establish relevance. As evidenced since (Daume and Marcu, 2006), Bayesian techniques have proven effective at this task: we construct a latent topic model on the basis of the document set and the query. This topic model effectively serves as a query expansion mechanism, which helps assess the relevance of individual sentences to the original query.

In recent years, three major techniques have emerged to perform multi-document summarization: graph-based methods such as LexRank (Erkan and Radev, 2004) for multi document summarization and Biased-LexRank (Otterbacher et al., 2008) for query focused summarization, language model methods such as KLSum (Haghighi and Vanderwende, 2009) and variants of KLSum based on topic models such as BayesSum (Daume and Marcu, 2006) and TopicSum (Haghighi and Vanderwende, 2009).

LexRank is a stochastic graph-based method for computing the relative importance of textual units in a natural text. The LexRank algorithm builds a weighted graph $G = (V, E)$ where each vertex in V is a linguistic unit (in our case sentences) and each weighted edge in E is a measure of similarity between the nodes. In our implementation, we model similarity by computing the cosine distance between the $TF \times IDF$ vectors representing each node. After the graph is generated, the PageRank algorithm *(*Page et al., 1999) is used to determine the most central linguistic units in the graph. To generate a summary we use the n most central lexical units, until the length of the target summary is reached. This method has no explicit control to avoid redundancy among the selected sentences, and the original algorithm does not address update or query-focused variants. Biased-LexRank (Otterbacher et al., 2008) makes LexRank sensitive to the query by introducing a prior belief about the ranking of the nodes in the graph, which reflects the similarity of sentences to the query. PageRank spreads the query similarity of a vertex to its close neighbors, so that we rank higher sentences that are similar to other sentences which are similar to the query. As a result, Biased-LexRank overcomes the lexical sparseness of the query and obtained state of the art results on the DUC 2005 dataset.

KLSum adopts a language model approach to compute relevance: the documents in the input set are modeled as a distribution over words (the original algorithm uses a unigram distribution over the bag of words in documents *D*). KLSum is a sentence extraction algorithm: it searches for a subset of the sentences in D with a unigram distribution as similar as possible to that of the overall collection D, but with a limited length. The algorithm uses Kullback-Lieber (KL) divergence $KL(P||Q) = \sum_w \log(\frac{P(w)}{Q(w)})P(w)$ to compute the similarity of the distributions. It searches for $S^* = \text{argmin}_{|S|<L} KL(P_D||P_S)$. This search is performed in a greedy manner, adding sentences one by one to S until the length L is reached, and choosing the best sentence as measured by KL-divergence at each step. The original method has no update or query focusing capability, but as a general modeling framework it is easy to adapt to a wide range of specific tasks.

TopicSum uses an LDA-like topic model (Blei *et al*. 2003) to classify words from a number of document sets (each set discussing a different topic) as either general non-content words, topic specific words and document specific word (this category refers to words that are specific to the writer and not shared across the document set). After the words are classified, the algorithm uses a KLSum variant to find the summary that best matches the unigram distribution of topic specific words. This method improves the results of

KLSum but it also has no update summary or query answering capabilities.

4 Dataset Collection

We now describe how we have constructed a dataset to evaluate QCFS algorithms, which we are publishing freely. We selected to build our dataset in the Consumer Health domain, a popular domain in the web (Cline and Haynes 2001) providing medical information at various levels of complexity, ranging from layman and up to expert information, because consumer health illustrates the need for exploratory search.

The PubMed repository, while primarily serving the academic community, is also used by laymen to ask health related questions. The PubMed query logs (Dogan et al., 2009) provide user queries with timestamps and anonymized user identification. They are publically available and include over 600K queries per day. In this dataset, Dogan and Murray found that query reformulation (typical of exploratory search) is quite frequent: "*In our dataset, 47% of all queries are followed by a new subsequent query. These users did not select any abstract or full text views from the result set. We make an operational assumption that these users' intent was to modify their search by reformulating their query.*" We used these logs to extract laymen queries relating to four topics: Asthma, Lung Cancer, Obesity and Alzheimer's disease. We extracted a single day query log. From these, we extracted sessions which contained the terms "*Asthma*", "*Lung Cancer*", "*Obesity*" or "*Alzheimer*". Sessions containing search tags (such as "[Author]") were removed to reduce the number of academic searches. The sessions were then manually examined and used to create zoom-in query chains of length 3 at most. The queries appear below:

Asthma:

Asthma causes→ asthma allergy→ asthma mold allergy;

Asthma treatment→asthma medication→corticosteroids;

Exercise induced asthma→ exercise for asthmatic;

Atopic dermatitis→ atopic dermatitis medications→ atopic dermatitis side effects;

Atopic dermatitis→ atopic dermatitis children→ atopic dermatitis treatment;

Atopic dermatitis → atopic dermatitis exercise activity → atopic dermatitis treatment;

Cancer:

Lung cancer→ lung cancer causes→ lung cancer symptoms;

Lung cancer diagnosis→ lung cancer treatment→lung cancer treatment side effects;

Stage of lung cancer→ lung cancer staging tests→ lung cancer TNM staging system;

Types of lung cancer→non-small cell lung cancer treatment→non-small cell lung cancer surgery;

Lung cancer in women→ risk factors for lung cancer in women→ treatment of lung cancer in women;

Lung cancer chemotherapy→ goals of lung cancer chemotherapy→ palliative care for lung cancer;

Obesity:

Salt obesity→retaining fluid;

Obesity screening→body mass index→BMI Validity;

Childhood obesity→childhood obesity low income→children diet and exercise;

Causes of childhood obesity→obesity and nutrition→school lunch;

Obesity and lifestyle change→obesity metabolism→superfoods antioxidant;

Obesity and diabetes→emergence of type 2 diabetes→type 2 diabetes and obesity in children;

Alzheimer's disease:

Alzheimer memory→helping retrieve memory alzheimer →alzheimer memory impairment nursing;

Cognitive impairment→Vascular Dementia→Vascular Dementia difference alzheimer;

Alzheimer's symptoms→alzheimer diagnosis→alzheimer medications;

Semantic dementia→first symptoms dementia→first symptoms alzheimer;

Figure 1: Queries Used to Construct Dataset

We asked medical experts to construct four document collections from well-known and reliable consumer health websites relating to the four subjects (Wikipedia, WebMD, and the NHS), so that they would provide general information relevant to the queries.

We then asked medical students to manually produce summaries of these four document collections for each query-chain. The medical students were instructed construct a text of up to 250 words that provides a good answer to each query in the chain. For each query in a chain the summarizers should assume that the person reading the summaries is familiar with the previous

summaries in the chain so they should avoid redundancy.

Three distinct human summaries were produced for each chain. For each chain, one summary was produced for each of the three queries, where the person producing the summary was not shown the next steps in the chain when answering the first query.

To simulate the exploratory search of the user we provided the annotators with a Solr[1] query interface for each document collection. The interface allowed querying the document set, reading the documents and choosing sentences which answer the query. After choosing the sentences, annotators can copy and edit the resulting summary in order to create an answer of up to 250 words. After processing the first two query chain summaries, the annotators held a post-hoc discussion about the different summaries in order to adjust their conception of the task.

The statistics on the collected dataset appear in the Tables below:

Document sets	# Docs	# Sentences	#Tokens / Unique
Asthma	125	1,924	19,662 / 2,284
Lung-Cancer	135	1,450	17,842 / 2,228
Obesity	289	1,615	21,561 / 2,907
Alzheimer's Disease	191	1,163	14,813 / 2,508

Queries	# Sessions	# Sentences	#Tokens / Unique
Asthma	5	15	36 / 14
Lung-Cancer	6	18	71 / 25
Obesity	6	17	45 / 29
Alzheimer's Disease	4	12	33 / 16

Manual Summaries	# Docs	# Sentences	#Tokens / Unique
Asthma	45	543	6,349 / 1,011
Lung-Cancer	54	669	8,287 / 1,130
Obesity	51	538	7,079 / 1,270
Alzheimer's Disease	36	385	5,031 / 966

Table 1: Collected Dataset Size Statistics

A key aspect of the dataset is that the same documents are summarized for each step of the chains, and we expect the summaries for each step to be different (that is, each answer is indeed responsive to the specific query it addresses). In addition, each answer is produced in the context of the previous steps, and only provides updated information with respect to previous answers. To ensure that the dataset indeed reflects these two aspects (responsiveness and freshness), we empirically verified that summaries created for advanced queries are different from the summaries created for the same queries by summarizers who did not see the previous summaries in the chain. We asked from additional annotators to create manual summaries of advanced queries from the query chain without ever seeing the queries from the beginning of the chain. For example, given the chain (*asthma causes* → *asthma allergy* → *asthma mold allergy*), we asked summarizers to produce an answer for the second query (*asthma allergy*) without seeing the first step, on the same input documents.

We used ROUGE to perform this validation: ROUGE compares a summary with a set of reference summaries and source documents. We first computed the mean ROUGE score of the second query summaries. The mean ROUGE score is the mean score of each manual summary *vs.* all other summaries about the same query. We got ($r1 = 0.52, r2 = 0.22, rs4 = 0.13$). The mean ROUGE scores of the same second query summaries by people who did not see the previous query were markedly lower: ($r1 = 0.40, r2 = 0.22, rs4 = 0.01$). We only verified the asthma dataset in this manner. The results, except for the R2 test, had statistically significant difference with 95% confidence interval. All the data, code and an annotated example can be found in our site[2].

We show below slightly shortened manual summaries created in answer to the following query chain: "*obesity screening -> body mass index -> BMI Validity*". We manually annotated the summaries to highlight how the exploration process develops and indicate some of the information retrieval challenges faced when addressing the task: underlined terms are directly relevant to the current query; ***bold italic terms*** have likely triggered the curiosity of the reader and led to asking the next query. In this example, a user asks about obesity screening, and reformulates his query using the technical term of BMI, which is prevalent in the produced answer.

Q1: obesity screening

Body mass index (BMI) is a measure used to determine

[1] http://lucene.apache.org/solr/

[2] http://www.cs.bgu.ac.il/~nlpproj/QCFS/dataset.html

childhood overweight and obesity. A child's weight status is determined using an age- and sex-specific percentile for ***BMI*** rather than the ***BMI*** categories used for adults because children's body composition varies as they age and varies between boys and girls. For children and adolescents (aged 2-19 years): Overweight is defined as a ***BMI*** at or above the 85th percentile and lower than the 95th percentile for children of the same age and sex. Obesity is defined as a ***BMI*** at or above the 95th percentile for children of the same age and sex. ***BMI*** is not a perfect measure, but rather a helpful indicator. It doesn't consider things like being muscular or having a larger than average body frame and because growth patterns vary greatly among children. As a result, doctors also factor a child's growth and development into consideration when screening for overweight and obesity. Obesity screening of adults using ***BMI*** may be less accurate for adults, due to the coarseness of the measurement: Heavily muscled athletes may have a high ***BMI***, as many women who enter menopause, who tend to gain weight due to increased water retention.

Q2: body mass index

The ***body mass index (BMI)***, or Quetelet ***index***, is a measure for human body shape based on an individual's mass and height. Devised in the mid-1800s by Adolphe Quetelet during the course of developing "social physics", it is defined as an individual's body mass divided by the square of their height, the units being kg/m^2. ***BMI*** was explicitly cited as being ***appropriate*** for population studies, and ***inappropriate*** for individual diagnosis. ***BMI*** provides a simple measure of a person's thickness, allowing health professionals to discuss over-weight and underweight problems more objectively with their patients. However, ***BMI*** has become ***controversial*** because many people, including physicians, have come to rely on its apparent ***authority*** for medical diagnosis. However, it was originally meant to be used as a simple means of classifying sedentary individuals, or rather, populations, with an average body composition. For these individuals, the current value settings are as follows: (...). Nick Korevaar (a mathematics lecturer from the University of Utah) suggests that instead of squaring the body height or cubing the body height, it would be more ***appropriate*** to use an exponent of between 2.3 and 2.7 (as originally noted by Quetelet).

Q3: BMI Validity

BMI has become controversial because many people, including physicians, have come to rely on its apparent numerical authority for medical diagnosis, but that was never the BMI's purpose; it is meant to be used as a simple means of classifying sedentary populations with an average body composition. In an article published in the July edition of 1972 of the Journal of Chronic Diseases, Ancel Keys explicitly cited BMI as being appropriate for population studies, but inappropriate for individual diagnosis. These ranges of BMI values are valid only as statistical categories While BMI is a simple, inexpensive method of screening for weight categories, it is not a good diagnostic tool: It does not take into account age, gender, or muscle mass. (...).

Figure 2: Query Chain Summary Annotated Example

5 Algorithms

In this section, we first explain how we adapted the previously mentioned methods to the QCFS task, thus producing 3 strong baselines. We then describe our new algorithm for QCFS.

5.1 Focused KLSum

We adapted KLSum to QCFS by introducing a simple document selection step in the algorithm. The method is: given a query step q, we first select a focused subset of documents from $D, D(q)$. We then apply the usual KLSum algorithm over $D(q)$. This approach does not make any effort to reduce redundancy from step to step in the query chain. In our implementation, we compute $D(q)$ by selecting the top-10 documents in D ranked by $TF \times IDF$ scores to the query, as implemented in SolR.

5.2 KL-Chain-Update

KL-Chain-Update is a slightly more sophisticated variation of KLSum that answers a query chain (instead a single query). When constructing a summary, we update the unigram distribution of the constructed summary so that it includes a smoothed distribution of the previous summaries in order to eliminate redundancy between the successive steps in the chain. For example, when we summarize the documents that were retrieved as a result to the first query, we calculate the unigram distribution in the same manner as we did in Focused KLSum; but for the second query, we calculate the unigram distribution as if all the sentences we selected for the previous summary were selected for the current query too, with a damping factor. In this variant, the Unigram Distribution estimate of word X is computed as:

$$\frac{\left(\text{Count}(W, CurrentSum) + \frac{\text{Count}(W, PreviousSum)}{SmoothingFactor}\right)}{\text{Length}(CurrentSum) + \frac{\text{Length}(\text{PreviousSum} \cap CurrentSum)}{SmoothingFactor}}$$

5.3 ChainSum

ChainSum is our adaptation of TopicSum to the QCFS task. We developed a novel Topic Model to identify words that are associated to the current query and not shared with the previous queries. We achieved this with the following model. For each query in a chain, we consider the documents D_c which are "good answers" to the query; and D_P which are the documents used to answer the previous steps of the chain. We assume in this model that these document subsets are observable (in our implementation, we select these subsets by ranking the documents for the query based on *TFxIDF* similarity).

1. G is the *general words* topic, it is intended to capture stop words and non-topic specific vocabulary. Its distribution φ_G is drawn for all the documents from $Dirichlet(V, \lambda_G)$.

2. S_i is the *document specific* topic; it represents words which are local for a specific document. ϕ_{S_i} is drawn for each document from $Dirichlet(V, \lambda_{S_i})$.

3. N is the *new content* topic, which should capture words that are characteristic for D_c. ϕ_N is drawn for all the documents in D_c from $Dirichlet(V, \lambda_N)$.

4. O captures *old content* from D_P, ϕ_O is drawn for all the documents in D_P from $Dirichlet(V, \lambda_O)$.

5. R captures *redundant information* between D_c and D_p, ϕ_R is drawn for all the documents in $D_p \cup D_c$ from $Dirichlet(V, \lambda_R)$.

6. For documents from D_c we draw from the distribution ψ_{t_1} over topics (G, N, R, S_i) from a Dirichlet prior with pseudo-counts (10.0,15.0,15.0,1.0)[3] . For each word in the document, we draw a topic Z from ψ_t, and a word W from the topic indicated by Z.

7. For documents from D_p, we draw from the distribution ψ_{t_2} over topics (G, O, R, S_i) from a Dirichlet prior with pseudo-counts (10.0,15.0,15.0,1.0) . The words are drawn in the same manner as in t_1.

8. For documents in $D \setminus (D_c \cup D_p)$ we draw from the distribution ψ_{t_3} over topics (G, S_i) from a Dirichlet prior with pseudo-counts (10.0,1.0) . The words are also drawn in the same manner as in t_1.

The plate diagram of this generative model is shown in Fig.3.

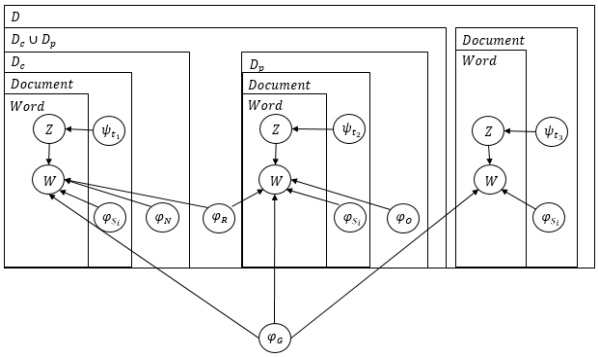


Figure 3 Plate Model for Our Topic Model

We implemented inference over this topic model using Gibbs Sampling (we distribute the code of the sampler together with our dataset). After the topic model is applied to the current query, we apply KLSum only on words that are assigned to the new content topic. Fig.4 summarizes the algorithm data flow.

When running this topic model on our dataset, we observe: D_c mean size was 978 words and 375 unique words. D_P mean size was 1374 words and 436 unique words. D_c and D_P mean on average 159 words. These figures show there is high lexical overlap between the summaries answering query qi and qi+1 and highlight the need to distinguish new and previously exposed content.

In the ChainSum model, the topic R aims at modeling redundant information between the previous summaries and the new summary. We intend in the future to exploit this information to construct a contrastive model of content selection. In the current version, R does not play an active role in content selection. We, therefore, tested a variant of ChainSum that did not include φ_R and obtained results extremely similar to the full model, which we report below.

[3] All pseudo-counts were selected empirically

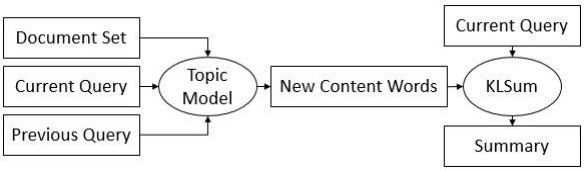


Figure 4 ChainSum Architecture

5.4 Adapted LexRank

In LexRank, the algorithm creates a graph where nodes represent the sentences from the text and weighted edges represent the cosine-distance of each sentence's *TFxIDF* vectors. After creating the graph, PageRank is run to rank sentences. We adapted LexRank to QCFS in two main ways: we extend the sentence representation scheme to capture semantic information and refine the model of sentences similarity so that it captures query answering instead of centrality. We tagged each sentence with Wikipedia terms using the Illinois Wikifier (Ratinov et al., 2011) and with UMLS (Bodenreider, 2004) terms using HealthTermFinder (Lipsky-Gorman and Elhadad, 2011). UMLS is a rich medical ontology, which is appropriate to the consumer health domain.

We changed the edges scoring formula to use the sum of Lexical Semantic Similarity (LSS) functions (Li et al., 2007) on lexical terms, Wikipedia terms and UMLS terms:

$$Score(U,V) = LSS_{lexical}(U,V) + a * LSS_{wiki}(U,V) + b * LSS_{UMLS}(U,V)$$

Where:

$$LSS(S_1,S_2) = \frac{\sum_i (MAX_j(\frac{Sim(W_i^1, W_j^2)}{Sim(W_i^1, W_i^1)}) IDF(W_i^1))}{\sum_i IDF(W_i^1)}$$

Instead of using the cosine distance, in order to incorporate advanced word/term similarity functions. For lexical terms, we used the identity function, for Wikipedia term we used Wikiminer (Milne, 2007), and for UMLS we used Ted Pedersen UMLS similarity function (McInnes et al., 2009). Finally, instead of PageRank, we used SimRank (Haveliwala, 2002) to identify the nodes most similar to the query node and not only the central sentences in the graph.

6 Evaluation

6.1 Evaluation Dataset

We worked on the dataset we created for QCFS and added semantic tags: 10% of the tokens had Wikipedia annotations and 33% had a UMLS annotation.

6.2 Results

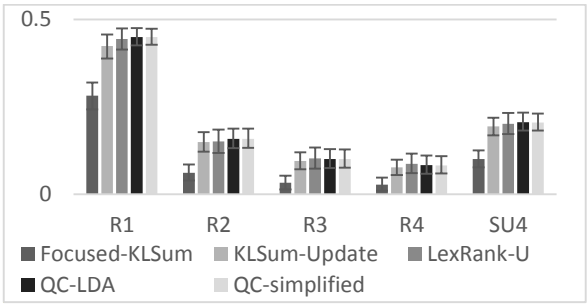


Figure 5: ROUGE Recall Scores (with stemming and stop-words)

For Focused KLSum we received ROUGE scores of (r1 = 0.281, r2 = 0.061, su4 = 0.100), KL-Chain-Update (r1 = 0.424, r2 = 0.149, su4 = 0.193), ChainSum (r1 = 0.44988, r2 = 0.1587, su4 = 0.20594), ChainSum with t Simplified Topic model (r1 = 0.44992, r2 = 0.15814, su4 = 0.20507) and for Modified-LexRank (r1 = 0.444, r2 = 0.151, su4 = 0.201). All of the modified versions of our algorithm performed better than Focused KLSum with more than 95% confidence.

7 Conclusions

We presented a new summarization task tailored for the needs of exploratory search system. This task combines elements of question answering by sentence extraction with those of update summarization.

The main contribution of this paper is the definition of a new summarization task that corresponds to exploratory search behavior and the contribution of a novel dataset containing human summaries. This dataset is annotated with Wikipedia and UMLS terms for over 30% of the tokens. We controlled that the summaries cover only part of the input document sets (and are, therefore, properly focused) and sensitive to the position of the queries in the chain.

Four methods were evaluated for the task. The baseline methods based on KL-Sum show a sig-

nificant improvement when penalizing redundancy with the previous summarization.

This paper concentrated on "zoom in" query chains, other user actions such as "zoom out" or "switch topic" were left to future work. This paper concentrated on "zoom in" query chains, other user actions such as "zoom out" or "switch topic" were left to future work. The task remains extremely challenging, and we hope the dataset availability will allow further research to refine our understanding of topic-sensitive summarization and redundancy control.

In future work, we will attempt to derive a task-specific evaluation metric that exploits the structure of the chains to better assess relevance, redundancy and contrast.

Acknowledgments

This work was supported by the Israeli Minister of Science (Grant #3-8705) and by the Lynn and William Frankel Center for Computer Sciences, Ben-Gurion University. We thank the reviewers for extremely helpful advice.

References

Marcia J. Bates. 1989. *The design of browsing and berrypicking techniques for the online search interface*, Online Information Review, 13(5), 407-424.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. *Latent dirichlet allocation*, the Journal of machine Learning research, 3, 993-1022.

Olivier Bodenreider. 2004. *The unified medical language system (UMLS): integrating biomedical terminology*, Nucleic acids research, 32(suppl 1), D267-D270.

John M. Conroy, Judith D. Schlesinger, and Dianne P. O'Leary. 2011. Nouveau-rouge: *A novelty metric for update summarization*, Computational Linguistics, 37(1), 1-8.

Rebecca JW Cline, and Katie M. Haynes. 2001. *Consumer health information seeking on the Internet: the state of the art*, Health education research, 16(6), 671-692.

Daume Hal and Daniel Marcu. 2006. *Bayesian query-focused summarization*, In Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics (pp. 305-312). Association for Computational Linguistics.

Jean-Yves Delort, and Enrique Alfonseca. 2012. *DualSum: a Topic-Model based approach for update summarization*, In Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (pp. 214-223). Association for Computational Linguistics.

Rezarta Islamaj Dogan, G. Craig Murray, Aurélie Névéol, and Zhiyong Lu. 2009. *Understanding PubMed® user search behavior through log analysis*, Database: The Journal of Biological Databases & Curation, 2009.

Günes Erkan, and Dragomir R. Radev. 2004. *LexRank: Graph-based lexical centrality as salience in text summarization*, J. Artif. Intell. Res.(JAIR), 22(1), 457-479.

Surabhi Gupta, Ani Nenkova, and Dan Jurafsky. 2007. *Measuring importance and query relevance in topic-focused multi-document summarization*, In Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions (pp. 193-196). Association for Computational Linguistics.

Aria Haghighi, and Lucy Vanderwende. 2009. *Exploring content models for multi-document summarization*, In Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (pp. 362-370). Association for Computational Linguistics.

Glen Jeh, and Jennifer Widom. 2002. *SimRank: a measure of structural-context similarity*, In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 538-543). ACM.

Baoli Li, Joseph Irwin, Ernest V. Garcia, and Ashwin Ram. 2007. *Machine learning based semantic inference: Experiments and Observations at RTE-3*, In Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing (pp. 159-164). Association for Computational Linguistics.

Chin-Yew Lin. 2004. *Rouge: A package for automatic evaluation of summaries*, In Text Summarization Branches Out: Proceedings of the ACL-04 Workshop (pp. 74-81).

Sharon Lipsky-Gorman, and Noémie Elhadad 2011. *ClinNote and HealthTermFinder: a pipeline for*

processing clinical notes, Columbia University Technical Report, Columbia University.

Gary Marchionini. 2006. *Exploratory search: from finding to understanding*, Communications of the ACM, 49(4), 41-46.

Bridget T. McInnes, Ted Pedersen, and Serguei VS Pakhomov. (2009). *UMLS-Interface and UMLS-Similarity: open source software for measuring paths and semantic similarity*, AMIA Annual Symposium Proceedings, American Medical Informatics Association.

David Milne. 2007. *Computing semantic relatedness using wikipedia link structure*, In Proceedings of the new zealand computer science research student conference.

Ani Nenkova, and Rebecca J. Passonneau. 2004. *Evaluating Content Selection in Summarization: The Pyramid Method*, In HLT-NAACL (pp. 145-152).

Jahna Otterbacher, Gunes Erkan, and Dragomir R. Radev. 2009. *Biased LexRank: Passage retrieval using random walks with question-based priors*, Information Processing & Management, 45(1), 42-54.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: bringing order to the web*,

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. *Local and Global Algorithms for Disambiguation to Wikipedia*, In ACL (Vol. 11, pp. 1375-1384).

Ryen W. White, and Resa A. Roth. 2009. *Exploratory search: Beyond the query-response paradigm. Synthesis Lectures on Information Concepts, Retrieval, and Service*s, 1(1), 1-98.

Exploiting Timelines to Enhance Multi-document Summarization

Jun-Ping Ng[1,2], **Yan Chen**[3], **Min-Yen Kan**[2,4], **Zhoujun Li**[3]

[1]DSO National Laboratories, Singapore
[2]School of Computing, National University of Singapore, Singapore
[3]State Key Laboratory of Software Development Environment, Beihang University, China
[4]Interactive and Digital Media Institute, National University of Singapore, Singapore
njunping@dso.org.sg

Abstract

We study the use of temporal information in the form of timelines to enhance multi-document summarization. We employ a fully automated temporal processing system to generate a timeline for each input document. We derive three features from these timelines, and show that their use in supervised summarization lead to a significant 4.1% improvement in ROUGE performance over a state-of-the-art baseline. In addition, we propose TIMEMMR, a modification to Maximal Marginal Relevance that promotes temporal diversity by way of computing time span similarity, and show its utility in summarizing certain document sets. We also propose a filtering metric to discard noisy timelines generated by our automatic processes, to purify the timeline input for summarization. By selectively using timelines guided by filtering, overall summarization performance is increased by a significant 5.9%.

1 Introduction

There has been a good amount of research invested into improving the temporal interpretation of text. Besides the increasing availability of annotation standards (e.g., TIMEML (Pustejovsky et al., 2003a)) and corpora (e.g., TIDES (Ferro et al., 2000), TimeBank (Pustejovsky et al., 2003b)), the community has also organized three successful evaluation workshops — TempEval-1 (Verhagen et al., 2009), -2 (Verhagen et al., 2010), and -3 (Uzzaman et al., 2013). As the state-of-the-art improves, these workshops have moved away from the piecemeal evaluation of individual temporal processing tasks and towards the evaluation of complete end-to-end systems in TempEval-3. We believe our understanding of the temporal information found in text is sufficiently robust, and that there is an opportunity to now leverage this information in downstream applications. In this paper, we present our work in incorporating the use of such temporal information into multi-document summarization.

The goal of multi-document summarization is to generate a summary which includes the main points from an input collection of documents with minimal repetition of similar points. We hope to improve the quality of the summaries that are generated by considering temporal information found in the input text. To motivate how temporal information can be useful in summarization, let us refer to Figure 1. The three sentences describe a recent cyclone and a previous one which happened in 1991. Recognizing that sentence (3) is about a storm that had happened in the past is important when writing a summary about the recent storm, as it is not relevant and can likely be excluded.

It is reasonable to expect that a collection of documents about the recent storm will contain more references to it, compared with the earlier one that happened in 1991. Visualized on a timeline, this will translate to more events (bolded in Figure 1) around the time when the recent storm occurred. There should be fewer events mentioned in the collection for the earlier 1991 time period. Figure 2 illustrates a possible timeline laid out with the events found in Figure 1. The events from the more recent storm are found together at the same time. There are fewer events which talk about the previous storm. Thus, temporal information does assist in identifying which sentences are more relevant to the final summary.

Our work is significant as it addresses an important gap in the exploitation of temporal information. While there has been prior work making use of temporal information for multi-document

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 923–933,
Baltimore, Maryland, USA, June 23-25 2014.

(1) A fierce cyclone **packing** extreme winds and torrential rain **smashed** into Bangladesh's southwestern coast Thursday, **wiping** out homes and trees in what officials **described** as the worst storm in years.
(2) More than 100,000 coastal villagers have been **evacuated** before the cyclone made landfall.
(3) The storm matched one in 1991 that **sparked** a tidal wave that **killed** an estimated 138,000 people, Karmakar told AFP.

Figure 1: Modified extract from a news article which describes a cyclone landfall. Several events which appear in Figure 2 are bolded.

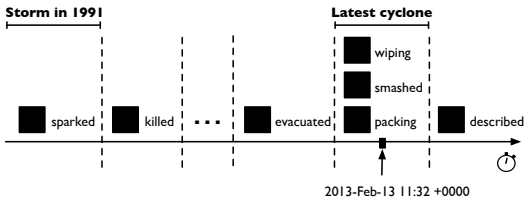


Figure 2: Possible timeline for events in Figure 1.

summarization, they 1) have been largely confined to helping to chronologically order content within summaries (Barzilay et al., 1999), or 2) focus only on the use of recency as an indicator of saliency (Goldstein et al., 2000; Wan, 2007). In this work we construct timelines (as a representation of temporal information) automatically and incorporate them into a state-of-the-art multi-document summarization system. This is achieved with 1) three novel features derived from timelines to help measure the saliency of sentences, as well as 2) TIMEMMR, a modification to the traditional Maximal Marginal Relevance (MMR) (Carbonell and Goldstein, 1998). TimeMMR promotes diversity by additionally considering temporal information instead of just lexical similarities. Through these, we demonstrate that temporal information is useful for multi-document summarization. Compared to a competitive baseline, significant improvements of up to 4.1% are obtained.

Automatic temporal processing systems are not perfect yet, and this may have an impact on their use for downstream applications. This work additionally proposes the use of the lengths of timelines as a metric to gauge the usefulness of timelines. Together with the earlier described contributions, this metric further improves summarization, yielding an overall 5.9% performance increase.

2 Related Work

Barzilay et al. (1999) were one of the first to use time for multi-document summarization. They recognized the importance of generating a summary which presents the time perspective of the summarized documents correctly. They estimated the chronological ordering of events with a small set of heuristics, and also made use of lexical patterns to perform basic time normalization on terms like "today" relative to the document creation time. The induced ordering is used to present the selected summary content, following the chronological order in the original documents.

In another line of work, Goldstein et al. (2000) made use of the temporal ordering of documents to be summarized. In computing the relevance of a passage for inclusion into the final summary, they considered the recency of the passage's source document. Passages from more recent documents are deemed to be more important. Wan (2007) and Demartini et al. (2010) made similar assumptions in their work on TIMEDTEXTRANK and entity summarization, respectively.

Instead of just considering the notion of recency, Liu et al. (2009) proposed an interesting approach using a temporal graph. Events within a document set correspond to vertices in their proposed graph, while edges are determined by the temporal ordering of events. From the resulting weakly-connected graph, the largest forests are assumed to contain key topics within the document set and used to influence a scoring mechanism which prefers sentences touching on these topics.

Wu (2008) also made use of the relative ordering of events. He assigned complete timestamps to events extracted from text. After laying out these events onto a timeline by making use of these timestamps, the number of events that happen within the same day is used to influence sentence scoring. The motivation behind this approach is that days which have a large number of events should be more important and more worthy of reporting than others.

These prior works target either 1) sentence re-ordering, or 2) the use of recency as an indicator of saliency. In sentence re-ordering, final summaries are re-arranged so that the extracted sentences that form the summary are in a chronological order. We argue that this may not be appropriate for all summaries. Depending on the style of writing or journalistic guidelines, a summary can arguably be written in a number of ways. The use of recency

as an indicator of saliency is useful, yet disregards other accessible temporal information. If a summary of a whole sequence of events is desired, recency becomes less useful.

The work of Wu (2008) is closely related to one of the features proposed in this paper. He had also made use of temporal information to weight sentences to generate summaries. However his approach is guided by the number of events happening within the same time span, and relies on event co-referencing. In this work, we have simplified this idea by dropping the need for event co-referencing (removing a source of propagated error), and augmented it with two additional features derived from timelines. By doing so, we are able to make better use of the available temporal information, taking into account all known events and the time in which they occur.

A useful note here is that this work is arguably different from the Temporal Summarization (TmpSum) track at the Text Retrieval Conference (Aslam et al., 2013). Given a large stream of data in real-time, the purpose of the TmpSum track is to look out for a query event, and retrieve specific details about the event over a period of time. Systems are also expected to identify the source sentences from which these details are retrieved. This is not the same as our approach here, which makes use of temporal information encoded in timelines to generate prose summaries.

3 Methodology

To incorporate temporal information into multi-document summarization, we adopt the workflow in Figure 3, which has two key processes: 1) temporal processing, and 2) summarization.

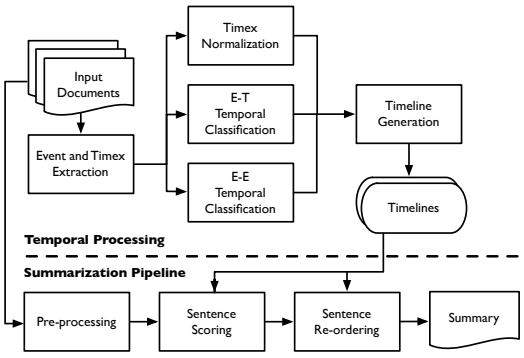


Figure 3: Incorporating temporal information into the SWING summarization pipeline.

Temporal Processing generates timelines from text, one for each input document. Timelines are well-understood constructs which have often been used to represent temporal information (Denis and Muller, 2011; Do et al., 2012). They indicate the temporal relationships between two basic temporal units: 1) events, and 2) time expressions (or timexes for short). In this work, we adopt the definitions proposed in the standardized TIMEML annotation (Pustejovsky et al., 2003a). An event refers to an eventuality, a situation that occurs or an action; while a timex is a reference to a particular date or time (e.g. "*2013 December 31*").

Following the "divide-and-conquer" approach described in Verhagen et al. (2010), results from the three temporal processing steps: 1) timex normalization, 2) event-timex temporal relationship classification, and 3) event-event temporal relationship classification, are merged to obtain timelines (top half of Figure 3). We tap on existing systems for each of these steps (Ng and Kan, 2012; Strötgen and Gertz, 2013; Ng et al., 2013).

Summarization. We make use of a state-of-the-art summarization system, SWING (Ng et al., 2012) (bottom half of Figure 3). SWING is a supervised, extractive summarization system which ranks sentences based on scores computed using a set of features in the *Sentence Scoring* phase. The Maximal Marginal Relevance (MMR) algorithm is then used in the *Sentence Re-ordering* phase to re-order and select sentences to form the final summary. The timelines built in the earlier temporal processing can be incorporated into this pipeline by deriving a set of features used to score sentences in *Sentence Scoring*, and as input to the MMR algorithm when computing similarity in *Sentence Re-ordering*.

3.1 Timelines from Temporal Processing

A typical timeline used in this work has been shown earlier in Figure 2. The arrowed, horizontal axis is the timeline itself. The timeline can be viewed as a continuum of time, with points on the timeline referring to specific moments of time. Small solid blocks on the timeline itself are references to absolute timestamps along the timeline (e.g., "*2013-Feb-13 11:32 +0000*" in the figure).

The black square boxes above the timeline denote events. Events can either occur at a specific instance of time (e.g., an explosion), or over a period of time (e.g. a football match). Generalizing, we refer to the time period an event takes place in as its *time span* (vertical dotted lines). As a simpli-

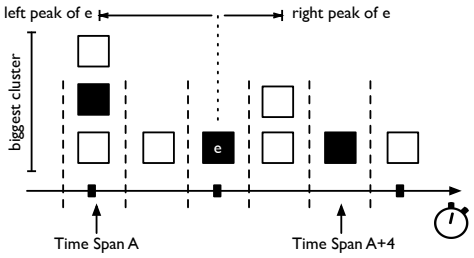


Figure 4: A simplified timeline illustrating how the various timeline features can be derived.

fying assumption, events are laid out on the timeline based on the starting time of their time span. Note that in our work, time spans may not correspond to specific instances of time, but instead help in inferring an ordering of events. Events which appear to the left of others take place earlier, while events within the same time span happen together over the same time period.

3.2 Sentence Scoring with Timelines

We derive three features from the constructed timelines, which are then used for downstream *Sentence Scoring*. Figure 4 shows a simplified timeline, along with annotations that are referenced in this section to help explain how these timeline features are derived.

1. Time Span Importance (TSI). We hypothesize that when more events happen within a particular time span, that time span is potentially more relevant for summarization. Sentences that mention events found in such a time span should be assigned higher scores. Referring to Figure 1, whose timeline is shown in Figure 2, we see that the time span with the most number of events is when the latest cyclone made landfall. Assigning higher scores for sentences which contain events in this time span will help us to select more relevant sentences if we want a summary about the cyclone.

Let TS_L be the time span with the largest number of events in a timeline. The importance of a time span TS_i is computed by normalizing the number of events in TS_i against the number of events in TS_L. The TSI of a sentence s is then the sum of the time span importance associated to all the words in s:

$$TSI(s) = \frac{\sum_{w \in s} \frac{|TS_w|}{|TS_L|}}{|s|} \quad (1)$$

where TS_w denotes the time span which a word w is associated with, and $|TS_w|$ is the number of events within the time span.

2. Contextual Time Span Importance (CTSI). The importance of a time span may not depend solely on the number of events that happen within it. If it is near time spans which are "important" (i.e., one that has a large number of events), it should also be of relative importance. A more concrete illustration of this can also be seen in Figure 1. Sentence (2) explains that a lot of people have been evacuated prior to the cyclone making landfall. It is imaginable that this can be useful information to be included in a summary, even though from looking at the corresponding timeline in Figure 2, the "*evacuated*" event falls in a time span with a low importance score (i.e., the time span only has one event). CTSI seeks to promote sentences such as this.

We derive the CTSI of a sentence by first computing the contextual importance of words in the sentence. We define the contextual importance of a word found in time span TS_i as a weighted sum of the time span importance of the two nearest peaks TS_{lp} and TS_{rp} found to the left and right of TS_i, respectively. In Figure 4, taking reference from event e (shaded in black), the left peak to the time span which e is in happens to be time span A, while the right peak is time span $A + 4$. The contribution of each peak to the weighted sum is decayed by its distance from TS_i. Formally, the contextual time span importance of a word w can be expressed as:

$$\zeta(w) = \alpha \left(\frac{I_{lp}}{|TS_w - TS_{lp}|} \right) \times (1 - \alpha) \left(\frac{I_{rp}}{|TS_{rp} - TS_w|} \right) \quad (2)$$

where TS_w is the time span associated with w. I_{lp} and I_{rp} are the time span importance of the peaks to the left and right of TS_w respectively, while $|TS_w - TS_{lp}|$ and $|TS_{rp} - TS_w|$ are the number of time spans between the left and right peaks of TS_w respectively. α balances the importance of the left and right peaks, intuitively set to 0.5. The CTSI of a sentence is computed as:

$$CTSI(s) = \frac{\sum_{e \in \mathbb{E}_s} \zeta(e)}{|\mathbb{E}_s|} \quad (3)$$

where $\mathbb{E}_s$ denotes the set of events words in s.

3. Sentence Temporal Coverage Density (TCD). We first define the *temporal coverage* of a sentence. This corresponds to the number of time spans that the events in a sentence talk about. Suppose a sentence contains events which are associated with time spans TS_a, TS_b, TS_c. The time spans are ordered in the sequence they appear on

the timeline. Then the temporal coverage of a sentence is defined as the number of time spans between the earliest time span TS_a and the latest time span TS_c. Referring to Figure 4, suppose a sentence contains the three events which have been shaded black. The temporal coverage in this case includes all the time spans from time span A to time span $A + 4$, inclusive.

The constraint on the number of sentences that can be included in a summary requires us to select compact sentences which contain as many relevant facts as possible. Traditional lexical measures may attempt to achieve this by computing the ratio of keyphrases to the number of words in a sentence (Gong and Liu, 2001). Stated equivalently, when two sentences are of the same length, if one contains more keyphrases, it should contain more useful facts. TCD parallels this idea with the use of temporal information, i.e. if two sentences are of the same temporal coverage, then the one with more events should carry more useful facts.

Formally, if a sentence s contains events $\mathbb{E}_s = \{e_1, \ldots, e_n\}$, where each event is associated with a time span TS_i, then TCD is computed using:

$$TCD(s) = \frac{|\mathbb{E}_s|}{|TS_n - TS_1|} \quad (4)$$

where $|\mathbb{E}_s|$ is the number of events found in s, and $|TS_n - TS_1|$ is the temporal coverage of s.

3.3 Enhancing MMR with TimeMMR

In the sentence re-ordering stage of the SWING pipeline, the iterative MMR algorithm is used to adjust the score of a candidate sentence, s. In each iteration, s is penalized if it is lexically similar to other sentences that have already been selected to form the eventual summary $S = \{s_1, s_2, \ldots\}$. The motivating idea is to reduce repeated information by preferring sentences which bring in new facts.

Incorporating temporal information can potentially improve this. In Figure 5, the sentences describe many events which took place within the same time span. They describe the destruction caused by a hurricane with trees uprooted and buildings blown away. A summary about the hurricane need not contain all of these sentences as they are all describing the same thing. However it is not trivial for the lexically-motivated MMR algorithm to detect that events like "*passed*", "*uprooted*" or "*damaged*" are in fact repetitive.

Thus, we propose further penalizing the score of s if it contains events that happen in similar time spans as those contained in sentences within S. We refer to this as TIMEMMR. Modifying the MMR equation from Ng et al. (2012):

$$TimeMMR(s) = Score(s) - \gamma R2(s, S) - (1 - \gamma)\mathcal{T}(s, S) \quad (5)$$

where $Score(s)$ is the score of s, S is the set of sentences already selected to be in the summary from previous iterations, and $R2$ is the predicted ROUGE-2 score of s with respect to the already selected sentences (S). γ is a weighting parameter which is empirically set to 0.9 after tuning over a development dataset. $\mathcal{T}$ is the proportion of events in s which happen in the same time span as another event in any other sentence in S. Two events are said to be in the same time span if one happens within the time period the other happens in. For example, an event that takes place in "*2014 June*" is said to take place within the year "*2014*".

While TIMEMMR is proposed here as an improvement over MMR, the premise is that incorporating temporal information can be helpful to minimize redundancy in summaries. In future work, one could apply it to other state-of-the-art lexical-based approaches including that of Hendrickx et al. (2009) and Celikyilmaz and Hakkani-Tur (2010). We also believe the same idea can be transplanted even to non-lexical motivated techniques such as the corpus-based similarity measure proposed by Xie and Liu (2008). We chose to use MMR here as a proof-of-concept to demonstrate the viability of such a technique, and to easily integrate our work into SWING.

3.4 Gauging Usefulness of Timelines

Temporal processing is imperfect. Together with the simplifying assumptions that were made in timeline construction, our generated timelines have errors which propagate into the summarization process. With this in mind, we selectively employ timelines to generate summaries only when we are confident of their accuracy. This can be done by computing a metric which can be used to decide whether or not timelines should be used for a particular input document collection. We refer to this as *reliability filtering*.

We postulate that the length of a timeline can serve as a simple reliability filtering metric. The intuition for this is that for longer timelines (which contain more events), possible errors are spread over the entire timeline, and do not overpower any useful signal that can be obtained from the timeline features outlined earlier. Errors are however

(1) An official in Barisal, 120 kilometres south of Dhaka, spoke of severe **destruction** as the 500 kilometre-wide mass of cloud **passed** overhead.
(2) "Many trees have been **uprooted** and houses and schools **blown** away," Mostofa Kamal, a district relief and rehabilitation officer, told AFP by telephone.
(3) "Mud huts have been **damaged** and the roofs of several houses **blown** off," said the state's relief minister, Mortaza Hossain.

Figure 5: Extract from a news article which describes several events (bolded) happening at the same time.

very easily propagated into summary generation for shorter timelines, leading to less useful results.

We incorporate this into our process as follows: given an input document collection (which consists of 10 documents), the average size of all the timelines for each of these 10 documents is computed. Only when this value is larger than a threshold value are the timelines used.

4 Experiments and Results

The proposed timeline features and TIMEMMR were implemented on top of SWING, and evaluated on the test documents from TAC-2011 (Owczarzak and Dang, 2011). SWING makes use of three generic features and two features targeted specifically at guided summarization. Since the focus of this paper is on multi-document summarization, we employ only the three generic features, i.e., 1) sentence position, 2) sentence length, and 3) interpolated n-gram document frequency in our experiments below. Summarization evaluation is done using ROUGE-2 (R-2) (Lin and Hovy, 2003), as it has previously been shown to correlate well with human assessment (Lin, 2004) and is often used to evaluate automatic text summarization.

The results obtained are shown in Table 1. In the table, each row refers to a specific summarization system configuration. We also show the results of two reference systems, CLASSY (Conroy et al., 2011) and POLYCOM (Zhang et al., 2011), as benchmarks. CLASSY and POLYCOM are top performing systems at TAC-2011 (ranked 2nd and 3rd by R-2 in TAC 2011, respectively; the full version of SWING was ranked 1st with a R-2 score of 0.1380). From these results, we can see that SWING is a very competitive baseline.

Rows 9 to 16 additionally incorporate our timeline reliability filtering. We assume that the various input document sets to be summarized are available at the time of processing. Hence in these experiments, the threshold for filtering is set to be the average of all the timeline sizes over the whole input dataset. In a production environment where this assumption may not hold, this threshold could be set by empirical tuning over a development set.

	Configuration	R-2	Sig
R	SWING	0.1339	NA
B1	CLASSY	0.1278	-
B2	POLYCOM	0.1227	**
Without Filtering			
1	SWING+TSI+CTSI+TCD	0.1394	*
2	SWING+TSI+CTSI	0.1372	-
3	SWING+TSI+TCD	0.1372	-
4	SWING+CTSI+TCD	0.1387	*
5	SWING+TSI+CTSI+TCD+TMMR	0.1389	-
6	SWING+TSI+CTSI+TMMR	0.1374	-
7	SWING+TSI+TCD+TMMR	0.1343	-
8	SWING+CTSI+TCD+TMMR	0.1363	-
With Filtering			
9	SWING+TSI+CTSI+TCD	0.1418	**
10	SWING+TSI+CTSI	0.1378	**
11	SWING+TSI+TCD	0.1389	**
12	SWING+CTSI+TCD	0.1401	**
13	SWING+TSI+CTSI+TCD+TMMR	0.1402	**
14	SWING+TSI+CTSI+TMMR	0.1397	**
15	SWING+TSI+TCD+TMMR	0.1376	*
16	SWING+CTSI+TCD+TMMR	0.1390	**

Table 1: R-2 scores after incorporating temporal information into SWING. '**' and '*' denotes significant differences with respect to Row R (paired one-tailed Student's t-test; $p < 0.05$ and $p < 0.1$ respectively), and TMMR denotes TIMEMMR.

Row 1 shows the usefulness of the proposed timeline-based features. A statistically significant improvement of 4.1% is obtained with the use of all three features over SWING. When we use reliability filtering (Row 9), this improvement increases to 5.9%.

The ablation test results in Rows 2 to 4 show a drop in R-2 each time a feature is left out. With the exception of Row 4, removing a feature lessens the improvement in R-2 to be insignificant from SWING's. The same drop occurs even when reliability filtering is used (Rows 9 to 12). These indicate that all the proposed features are important and need to be used together to be effective.

Rows 5 to 8 and Rows 13 to 16 show the effect of TIMEMMR. While the results do not uniformly show that TIMEMMR is effective, it can be helpful, such as when comparing Rows 2 and 6, or Rows 10 and 14, where R-2 improves marginally.

Looking at Rows 1 to 8, and Rows 9 to 16, we see the importance of reliability filtering. It is able

to guide the use of timelines such that significant improvements in R-2 over SWING are obtained.

To help visualize what the differences in these ROUGE scores mean, Figure 7 shows two summaries[1] generated for document set D1117C of the TAC-2011 dataset. The left one is produced by the configuration in Row 9, and the right one is produced by SWING without the use of any temporal information.

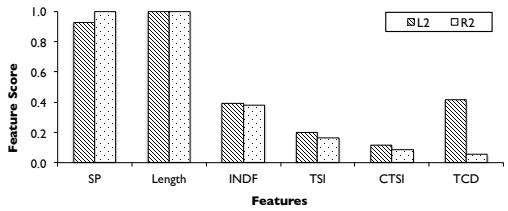


Figure 6: Breakdown of raw feature scores for sentences ($\mathbb{L}2$) and ($\mathbb{R}2$) from Figure 7.

The higher R-2 score obtained by the summary on the left (0.0873) compared to the one on the right (0.0723) suggests that temporal information can help to identify salient sentences more accurately. A closer look at sentences ($\mathbb{L}2$) and ($\mathbb{R}2$) and their R-2 scores (0.0424 and 0.0249, respectively) is instructive. Figure 6 shows the raw feature scores of both sentences. Both sentences score similarly for the SWING features of sentence position (SP), sentence length (Length), and interpolated n-gram document frequency (INDF); however, the scores for all three timeline features higher for ($\mathbb{L}2$) than ($\mathbb{R}2$). This helps our time sensitive system prefer ($\mathbb{L}2$).

5 Discussion

We now examine the proposed 1) timeline features, 2) TIMEMMR algorithm, and 3) reliability filtering metric in greater detail to gain insight into their efficacy. For the analysis on timeline features, we only present an analysis for TSI and CTSI due to space constraints.

Time Span Importance. Figure 8 shows the last sentences from a pair of summaries generated with and without the use of TSI (all other sentences were the same). The original articles describe an accident where casualties were suffered when a crane toppled onto a building. It is easy to see why ($\mathbb{L}1$) scores higher for R-2 — it describes the cause of the accident just as it occurred. ($\mathbb{R}1$) however talks about events which happened before the accident itself (e.g., how much of the tower had already been erected). In this case time span importance is able to correctly guide summary generation by favoring time spans containing events related to the actual toppling.

Contextual Time Span Importance. CTSI recognizes that events which happen around the time of a big cluster of other events can be important too. The benefits of this feature can be clearly seen in Figure 9. The summary on the left achieved a R-2 score of 0.1215 while the one on the right achieved 0.0861. ($\mathbb{L}2$) and ($\mathbb{L}3$) were both boosted by the use of the contextual importance feature.

Figure 10 shows an extract of the timeline generated for the source document from which ($\mathbb{L}3$) is extracted. The two events inside ($\mathbb{L}3$) fall in time spans A and B marked in the figure. Their proximity to the peak P between them gives the sentence a higher score for CTSI. This boost results in the sentence being selected for inclusion in the final summary. It turns out that this sentence was lifted exactly in one of the model summaries for this document set, resulting in a very good R-2 score when contextual importance is used.

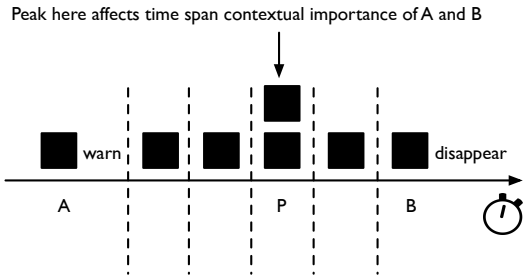


Figure 10: Extract of timeline generated for document APW_ENG_20070615.0356 from the TAC-2011 dataset.

Is TIMEMMR Useful? The experimental results do not conclusively affirm the usefulness of TIMEMMR. However we believe it is because the ROUGE measures that are used for evaluation are not suited for this purpose. Recall that TIMEMMR seeks to eliminate redundancy based on time span similarities and not lexical likeness. ROUGE, however, measures the latter.

An interesting case in point is given in Figure 11. The summary on the left is generated using TIMEMMR and achieved a lower ROUGE score. The one on the right is generated without TIMEMMR and scores higher, suggesting that TIMEMMR is not helpful. The key difference in

[1]The produced summaries are truncated to fit within a 100-word limit imposed by the TAC-2011 guidelines.

R-2: **0.0873**		R-2: **0.0723**
(𝕃1,ℝ1) The Army's surgeon general criticized stories in The Washington Post disclosing problems at Walter Reed Army Medical Center, saying the series unfairly characterized the living conditions and care for soldiers recuperating from wounds at the hospital's facilities.		
(𝕃2) Defense Secretary Robert Gates says people found to have been responsible for allowing substandard living conditions for soldier outpatients at Walter Reed Army Medical Center in Washington will be "held accountable," although so far no one in the Army chain of command has offered to resign.	$\neq\neq$	(ℝ2) A top Army general vowed to personally oversee the upgrading of Walter Reed Army Medical Center's Building 18, a dilapidated former hotel that houses wounded soldiers as outpatients.
(𝕃3) Top Army officials visited Building 18, the decrepit former hotel housing more than 80 recovering soldiers, outside	$\neq\neq$	(ℝ3) "I'm not sure it was an accurate representation," Lt. Gen. Kevin Kiley, chief of the Army Medical Command which oversees Walter Reed and all Army health care, told reporters during a news conference.
	$>>$	(ℝ4) The Washington

Figure 7: Generated summaries for document set D1117C from the TAC-2011 dataset. Left summary is generated by SWING+TSI+CTSI+TCD with filtering; right summary is by SWING.

R-2: **0.1683**		R-2: **0.1533**
.....		
(𝕃1) A piece of steel fell and sheared off one of the ties holding it to the building, causing it to detach and topple, said Stephen Kaplan	$\neq\neq$	(ℝ1) About 19 of the 44 stories of the crane had been erected and it was to be extended when a piece of steel fell and sheared

Figure 8: Extract from summaries for document set D1137G from the TAC-2011 dataset. Left extract is generated by SWING+TSI+CTSI+TCD; right extract is by SWING+CTSI+TCD.

the two summaries is (ℝ3). (𝕃3) is the equivalent of (ℝ4), while (𝕃4) is the full version of the truncated (ℝ5). TIMEMMR penalizes (ℝ3). (ℝ3) reports that the shoe-throwing incident happened as the U.S. President Bush appeared together with the Iraqi Prime Minister Nouri al-Maliki. However their joint appearance is already reported in (ℝ1) (and similarly (𝕃1)). (ℝ3) repeats what had been presented earlier. Since (ℝ1) and (ℝ3) talk about the same time span, TIMEMMR down-weights (ℝ3). We argue that this is better even though the ROUGE scores indicate otherwise. In future work it will be worthwhile to consider the use of metrics like Pyramid (Passonneau et al., 2005) which are less bound to superficial lexicons.

Reliability Filtering. Table 2 shows the effect of varying the filtering threshold on R-2 for the best performing configuration from Table 1 (i.e., SWING+TSI+CTSI+TCD). The result obtained in Row 9 using a threshold of 42.68 is also re-produced for reference. **T**=0 means that timelines are used for all input document sets, whereas **T**=100 means that no timelines are used, as the length of the longest timeline is less than 100.

As the threshold increases from 0 to 40–50, summarization performance improves while the number of document sets where temporal information is used is reduced. This suggests that filtering is successful in identifying timelines that are not sufficiently accurate to be useful for summarization. R-2 performance peaks around a threshold of 40. This affirms our use of the average length of timelines as the threshold value in our earlier experiments. Beyond 60, the R-2 scores are still higher than that obtained by SWING, but no longer significantly different. At these higher thresholds, temporal information is still able to help get an improvement in R-2. However as this affects only very few out of the 44 document sets, statistical variances mean that these R-2 scores are no longer

T	**R-2**	**Sig**	**#**	**T**	**R-2**	**Sig**	**#**
0	0.1394	*	44	**50**	0.1386	**	13
10	0.1382	-	43	**60**	0.1361	*	7
20	0.1377	-	41	**70**	0.1351	-	3
30	0.1393	**	35	**80**	0.1351	-	2
40	0.1426	**	22	**90**	0.1353	-	1
42.68	*0.1418*	**	*21*	**100**	0.1339	-	0

Table 2: Effect of different reliability filtering thresholds for SWING+TSI+CTSI+TCD. '**T**' is the threshold used; '**#**' is the number of input collections (out of 44) where timelines are used; '**' and '*' is statistical significance over SWING of $p < 0.05$ and $p < 0.1$, respectively.

R-2: **0.1215**		R-2: **0.0861**
(($\mathbb{L}$1,$\mathbb{R}$1) Caribbean coral species essential to the region's reef ecosystems are at risk of extinction as a result of climate change.		
($\mathbb{L}$2) But destructive fishing methods and overharvesting have reduced worldwide catches by 90 percent in the past two decades.	$\neq\neq$	($\mathbb{R}$2) The Coral Reef Task Force, created in the Clinton administration, regularly assesses coral health.
($\mathbb{L}$3) Scientists warn that up to half of the world's coral reefs could disappear by 2045.	$\neq\neq$	($\mathbb{R}$3) With a finished necklace retailing for up to 20,000 dollars (15,000 euros), red corals are among the world's most expensive wildlife commodities.
. . . . .		

Figure 9: Extract from summaries for document set D1131F from the TAC-2011 dataset. Left extract is generated by SWING+TSI+CTSI+TCD; right extract is by SWING+TSI+TCD.

R-2: **0.2643**		R-2: **0.2772**
($\mathbb{L}$1,$\mathbb{R}$1) – An Iraqi reporter threw his shoes at visiting U.S. President George W. Bush and called him a "dog" in Arabic during a news conference with Iraqi Prime Minister Nuri al-Maliki in Baghdad		
($\mathbb{L}$2,$\mathbb{R}$2) "All I can report is it is a size 10,.		
($\mathbb{L}$3) Muntadhar al-Zaidi, reporter of Baghdadiya television jumped and threw his two shoes one by one at the president, who ducked and thus narrowly missed being struck, raising chaos in the hall in Baghdad's heavily fortified green Zone.	$\neq\neq$	($\mathbb{R}$3) The incident occurred as Bush was appearing with Iraqi Prime Minister Nouri al-Maliki.
($\mathbb{L}$4) The president lowered his head and the first shoe hit the American and Iraqi flags behind the two leaders.	$\neq\neq$	($\mathbb{R}$4) Muntadhar al-Zaidi, reporter of Baghdadiya television jumped and threw his two shoes one by one at the president, who ducked and thus narrowly missed being struck, raising chaos in the hall in Baghdad's heavily fortified green Zone.
($\mathbb{L}$5) The	$\neq\neq$	($\mathbb{R}$5) The president lowered his head and the

Figure 11: Summaries for document set D1126E from the TAC-2011 dataset. Left summary is generated by SWING+TSI+CTSI+TCD+TIMEMMR; right summary is by SWING+TSI+CTSI+TCD.

significant from that produced by SWING.

6 Conclusion

We have shown in this work how temporal information in the form of timelines can be incorporated into multi-document summarization. We achieve this through two means, using: 1) three novel features derived from timelines to measure the saliency of sentences, and 2) TIMEMMR which considers time span similarity to enhance the traditional MMR's lexical diversity measure.

To overcome errors propagated from the underlying temporal processing systems, we proposed a reliability filtering metric which can be used to help decide when temporal information should be used for summarization. The use of this metric leads to an overall 5.9% gain in R-2 over the competitive SWING baseline.

In future work, we are keen to study our proposed timeline-related features more intrinsically in the context of human-generated summaries. This can help us better understand their value in improving content selection. As noted earlier, it will be also be useful to repeat our experiments with less lexicon-influenced measures like the Pyramid method (Passonneau et al., 2005). Manual assessment of the generated summaries can also be done to give a better picture of the quality of the summaries generated with the use of timelines. Finally, given the importance of reliability filtering, a natural question is if there are other metrics that can be used to get better results.

Acknowledgments

This research is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office.

This work is also partially supported by the National Natural Science Foundation of China (Grant Nos. 61170189, 61370126, 61202239), the Fund of the State Key Laboratory of Software Development Environment (Grant No. SKLSDE-2013ZX-19), and the Innovation Foundation of Beihang University for Ph.D. Graduates (YWF-13-T-YJSY-024).

References

Javed Aslam, Matthew Ekstrand-Abueg, Virgil Pavlu, Fernando Diaz, and Tetsuya Sakai. 2013. TREC 2013 Temporal Summarization. In *Proceedings of the 22nd Text Retrieval Conference (TREC)*, November.

Regina Barzilay, Kathleen McKeown, and Michael Elhadad. 1999. Information Fusion in the Context of Multi-document Summarization. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics (ACL)*, pages 550–557, June.

Jaime Carbonell and Jade Goldstein. 1998. The Use of MMR, Diversity-based Reranking for Reordering Documents and Producing Summaries. In *Proceedings of the 21st Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 335–336, August.

Asli Celikyilmaz and Dilek Hakkani-Tur. 2010. A Hybrid Hierarchical Model for Multi-document Summarization. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 815–824, July.

John M. Conroy, Judith D. Schlesinger, Jeff Kubina, Peter A. Rankel, and Dianne P. O'Leary. 2011. CLASSY 2011 at TAC: Guided and Multi-lingual Summaries and Evaluation Metrics. In *Proceedings of the Text Analysis Conference (TAC)*, November.

Gianluca Demartini, Malik Muhammad Saad Missen, Roi Blanco, and Hugo Zaragoza. 2010. Entity Summarization of News Articles. In *Proceedings of the 33rd Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 798–796, July.

Pascal Denis and Philippe Muller. 2011. Predicting Globally-Coherent Temporal Structures from Texts via Endpoint Inference and Graph Decomposition. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, July.

Quang Xuan Do, Wei Lu, and Dan Roth. 2012. Joint Inference for Event Timeline Construction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP)*, pages 677–689, July.

Lisa Ferro, Laurie Gerber, Inderjeet Mani, Beth Sundheim, and George Wilson. 2000. Instruction Manual for the Annotation of Temporal Expressions. Technical report, The MITRE Corporation.

Jade Goldstein, Vibhu Mittal, Jaime Carbonell, and Mark Kantrowitz. 2000. Multi-document Summarization by Sentence Extraction. In *Proceedings of the 2000 NAACL-ANLP Workshop on Automatic Summarization*, volume 4, pages 40–48, April.

Yihong Gong and Xin Liu. 2001. Generic Text Summarization Using Relevance Measure and Latent Semantic Analysis. In *Proceedings of the 24th Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 19–25, September.

Iris Hendrickx, Walter Daelemans, Erwin Marsi, and Emiel Krahmer. 2009. Reducing Redundancy in Multi-document Summarization using Lexical Semantic Similarity. In *Proceedings of the Workshop on Language Generation and Summarisation (UCNLG+Sum)*, pages 63–66, August.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL)*, volume 1, pages 71–78, May.

Chin-Yew Lin. 2004. Looking for a Few Good Metrics: ROUGE and its Evaluation. In *Working Notes of the 4th NTCIR Workshop Meeting*, June.

Maofu Liu, Wenjie Li, and Huijun Hu. 2009. Extractive Summarization Based on Event Term Temporal Relation Graph and Critical Chain. In *Information Retrieval Technology*, volume 5839 of *Lecture Notes in Computer Science*, pages 87–99. Springer Berlin Heidelberg.

Jun-Ping Ng and Min-Yen Kan. 2012. Improved Temporal Relation Classification using Dependency Parses and Selective Crowdsourced Annotations. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 2109–2124, December.

Jun-Ping Ng, Praveen Bysani, Ziheng Lin, Min-Yen Kan, and Chew-Lim Tan. 2012. Exploiting Category-Specific Information for Multi-Document Summarization. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 2093–2108, December.

Jun-Ping Ng, Min-Yen Kan, Ziheng Lin, Wei Feng, Bin Chen, Jian Su, and Chew-Lim Tan. 2013. Exploiting Discourse Analysis for Article-Wide Temporal Classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 12–23, October.

Karolina Owczarzak and Hoa Dang. 2011. Overview of the TAC 2011 Summarization Track: Guided Task and AESOP Task. In *Proceedings of the Text Analysis Conference (TAC)*, November.

Rebecca J. Passonneau, Ani Nenkova, Kathleen McKeown, and Sergey Sigelman. 2005. Applying the Pyramid Method in DUC 2005. In *Proceedings of the Document Understanding Conference Workshop on Text Summarization*, October.

James Pustejovsky, José Castano, Robert Ingria, Roser Saurí, Robert Gaizauskas, Andrea Setzer, and Graham Katz. 2003a. TimeML: Robust Specification of Event and Temporal Expressions in Text. In *Proceedings of the 5th International Workshop on Computational Semantics (IWCS)*, January.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, and Marcia Lazo. 2003b. The TIMEBANK corpus. In *Proceedings of Corpus Linguistics*, pages 647–656, March.

Jannik Strötgen and Michael Gertz. 2013. Multilingual and Cross-domain Temporal Tagging. *Language Resources and Evaluation*, 47(2):269–298.

Naushad Uzzaman, Hector Llorens, Leon Derczynski, Marc Verhagen, James F. Allen, and James Pustejovsky. 2013. SemEval-2013 Task 1: TEMPEVAL-3: Evaluating Time Expressions, Events, and Temporal Relations. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval)*, June.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Jessica Moszkowicz, and James Pustejovsky. 2009. The TempEval Challenge: Identifying Temporal Relations in Text. *Language Resources and Evaluation*, 43(2):161–179.

Marc Verhagen, Roser Saurí, Tommaso Caselli, and James Pustejovsky. 2010. SemEval-2010 Task 13: TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval)*, pages 57–62, July.

Xiaojun Wan. 2007. TimedTextRank: Adding the Temporal Dimension to Multi-Document Summarization. In *Proceedings of the 30th Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 867–868, July.

Mingli Wu. 2008. *Investigations on Temporal-Oriented Event-Based Extractive Summarization*. Ph.D. thesis, Hong Kong Polytechnic University.

Shasha Xie and Yang Liu. 2008. Using Corpus and Knowledge-based Similarity Measure in Maximum Marginal Relevance for Meeting Summarization. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 4985–4988, March.

Renxian Zhang, You Ouyang, and Wenjie Li. 2011. Guided Summarization with Aspect Recognition. In *Proceedings of the Text Analysis Conference (TAC)*, November.

A chance-corrected measure of inter-annotator agreement for syntax

Arne Skjærholt
Language technology group, dept. of informatics
University of Oslo
arnskj@ifi.uio.no

Abstract

Following the works of Carletta (1996) and Artstein and Poesio (2008), there is an increasing consensus within the field that in order to properly gauge the reliability of an annotation effort, chance-corrected measures of inter-annotator agreement should be used. With this in mind, it is striking that virtually all evaluations of syntactic annotation efforts use uncorrected parser evaluation metrics such as bracket F_1 (for phrase structure) and accuracy scores (for dependencies).

In this work we present a chance-corrected metric based on Krippendorff's α, adapted to the structure of syntactic annotations and applicable both to phrase structure and dependency annotation without any modifications. To evaluate our metric we first present a number of synthetic experiments to better control the sources of noise and gauge the metric's responses, before finally contrasting the behaviour of our chance-corrected metric with that of uncorrected parser evaluation metrics on real corpora.[1]

1 Introduction

It is a truth universally acknowledged that an annotation task in good standing be in possession of a measure of inter-annotator agreement (IAA). However, no such measure is in widespread use for the task of syntactic annotation. This is due to a mismatch between the formulation of the agreement measures, which assumes that the annotations have no or relatively little internal structure, and syntactic annotation where structure is the entire point of the annotation. For this reason efforts to gauge the quality of syntactic annotation are hampered by the need to fall back to simple accuracy measures. As shown in Artstein and Poesio (2008), such measures are biased in favour of annotation schemes with fewer categories and do not account for skewed distributions between classes, which can give high observed agreement, even if the annotations are inconsistent.

In this article we propose a family of chance-corrected measures of agreement, applicable to both dependency- and constituency-based syntactic annotation, based on Krippendorff's α and tree edit distance. First we give an overview of traditional agreement measures and why they are insufficient for syntax, before presenting our proposed metrics. Next, we present a number of synthetic experiments performed in order to find the best distance function for this kind of annotation; finally we contrast our new metric and simple accuracy scores as applied to real-world corpora before concluding and presenting some potential avenues for future work.

1.1 Previous work

The definitive reference for agreement measures in computational linguistics is Artstein and Poesio (2008), who argue forcefully in favour of the use of chance-corrected measures of agreement over simple accuracy measures. However, most evaluations of syntactic treebanks use simple accuracy measures such as bracket F_1 scores for constituent trees (NEGRA, Brants, 2000; TIGER, Brants and Hansen, 2002; Cat3LB, Civit et al., 2003; The Arabic Treebank, Maamouri et al., 2008) or labelled or unlabelled attachment scores for dependency syntax (PDT, Hajič, 2004; PCEDT Mikulová and Štěpánek, 2010; Norwegian Dependency Treebank, Skjærholt, 2013). The only work we know of using chance-corrected metrics

[1] The code used to produce the data in this paper, and some of the datasets used, are available to download at https://github.com/arnsholt/syn-agreement/

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 934–944,
Baltimore, Maryland, USA, June 23-25 2014.

is Ragheb and Dickinson (2013), who use MASI (Passonneau, 2006) to measure agreement on dependency relations and head selection in multi-headed dependency syntax, and Bhat and Sharma (2012), who compute Cohen's κ (Cohen, 1960) on dependency relations in single-headed dependency syntax. A limitation of the first approach is that token ID becomes the relevant category for the purposes of agreement, while the second approach only computes agreements on relations, not on structure.

In grammar-driven treebanking (or parsebanking), the problems encountered are slightly different. In HPSG and LFG treebanking annotators do not annotate structure directly. Instead, the grammar parses the input sentences, and the annotator selects the correct parse (or rejects all the candidates) based on discriminants[2] of the parse forest. In this context, de Castro (2011) developed a variant of κ that measures agreement over discriminant selection. This is different from our approach in that agreement is computed on annotator decisions rather than on the treebanked analyses, and is only applicable to grammar-based approaches such as HPSG and LFG treebanking.

The idea of using edit distance as the basis for an inter-annotator agreement metric has previously been explored by Fournier (2013). However that work used a boundary edit distance as the basis of a metric for the task of text segmentation.

1.2 Notation

In this paper, we mostly follow the notation and terminology of Artstein and Poesio (2008), with some additions. The key components in an agreement study are the *items* annotated, the *coders* who make judgements on individual items, and the *annotations* created for the items. We denote these as follows:

- The set of items $I = \{i_1, i_2, \dots\}$
- The set of coders $C = \{c_1, c_2, \dots\}$
- The set of annotations X is a set of sets $X = \{X_i | i \in I\}$ where each set $X_i = \{x_{ic} | c \in C\}$ contains the annotations for each item. If not all coders annotate all items, the different X_i will be of different sizes.

[2] A discriminant is an attribute of the analyses produced by the grammar where some of the analyses differ, e.g. is the word *jump* a noun or a verb, or does a PP attach to a VP or the VP's object NP.

In the case of nominal categorisation we will also use the set K of possible categories.

2 The metric

The most common metrics used in computational linguistics are the metrics κ (Cohen, 1960, introduced to computational linguistics by Carletta, 1996) and π (Scott, 1955). These metrics express agreement on a nominal coding task as the ratio $\kappa, \pi = {A_o - A_e}/{1 - A_e}$ where A_o is the observed agreement and A_e the expected agreement according to some model of "random" annotation. Both metrics have essentially the same model of expected agreement:

$$A_e = \sum_{k \in K} P(k|c_1)P(k|c_2) \quad (1)$$

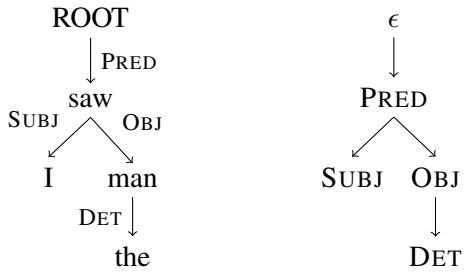
differing only in how they estimate the probabilities: κ assigns separate probability distributions to each coder based on their observed behaviour, while π uses the same distribution for both coders based on their aggregate behaviour.

Now, if we want to perform this same kind of evaluation on syntactic annotation it is not possible to use κ or π directly. In the case of dependency-based syntax we could conceivably use a variant of these metrics by considering the ID of a token's head as a categorical variable (the approach taken in Ragheb and Dickinson, 2013), but we argue that this is not satisfactory. This use of the metrics would consider agreement on categories such as "tokens whose head is token number 24", which is obviously not a linguistically informative category. Thus we have to reject this way of assessing the reliability of dependency syntax annotation. Also, this approach is not directly generalisable to constituency-based syntax.

For dependency syntax we could generalise these metrics similarly to how κ is generalised to κ_w to handle partial credit for overlapping annotations. Let the function $\text{LAS}(t_1, t_2)$ be the number of tokens with the same head and label in the two trees t_1 and t_2, $T(i)$ the set of trees possible for an item $i \in I$, and tokens the number of tokens in the corpus. Then we can compute an expected agreement as follows:

$$A_e = \frac{1}{\text{tokens}} \sum_{i \in I} \sum_{t_1, t_2 \in T(i)^2} \text{LAS}_e(t_1, t_2) \quad (2)$$

$$\text{LAS}_e(t_1, t_2) = P(t_1|c_1)P(t_2|c_2)\text{LAS}(t_1, t_2)$$



(a) The original dependency tree

(b) The tree used in comparisons

Figure 1: Transformation of dependency trees before comparison

We see three problems with this approach. First of all the number of possible trees for a sentence grows exponentially with sentence length, which means that explicitly iterating over all possible such pairs is computationally intractable, nor have we been able to easily derive an algorithm for this particular problem from standard algorithms.

Second, the question of which model to use for $P(t|c)$ is not straightforward. It is possible to use generative parsing models such as PCFGs or the generative dependency models of Eisner (1996), but agreement metrics require a model of *random* annotation, and as such using models designed for parsing runs the risk of over-estimating A_e, resulting in artificially low agreement scores.

Finally, it may be hard to establish a consensus in the field of which particular metric to use. As shown by the existence of three different metrics (κ, π and S (Bennett et al., 1954)) for the relatively simple task of nominal coding, the choice of model for $P(t|c)$ will not be obvious, and thus differing choices of generative model as well as different choices for parameters such as smoothing will result in subtly different agreement metrics. The results of these different metrics will not be directly comparable, which will make the results of groups using different metrics unnecessarily hard to compare.

Instead, we propose to use an agreement measure based on Krippendorff's α (Krippendorff, 1970; Krippendorff, 2004) and tree edit distance. In this approach we compare tree structures directly, which is extremely parsimonious in terms of assumptions, and furthermore sidesteps the problem of probabilistically modelling annotators' behaviour entirely. Krippendorff's α is not as commonly used as κ and π, but it has the advantage of being expressed in terms of an arbitrary *distance function* δ.

A full derivation of α is beyond the scope of this article, and we will simply state the formula used to compute the agreement. Krippendorff's α is normally expressed in terms of the ratio of observed and expected disagreements: $\alpha = 1 - {}^{D_o}/_{D_e}$, where D_o is the mean squared distance between annotations of the same item and D_e the mean squared distance between all pairs of annotations:

$$D_o = \sum_{i \in I} \frac{1}{|X_i| - 1} \sum_{c \in C} \sum_{c' \in C} \delta(x_{ic}, x_{ic'})^2$$

$$D_e = \frac{1}{\sum_{i \in I} |X_i| - 1} \sum_{i \in I} \sum_{c \in C} \sum_{i' \in I} \sum_{c' \in C} \delta(x_{ic}, x_{i'c'})^2$$

Note that in the expression for D_e, we are computing the difference between annotations for *different* items; thus, our distance function for syntactic trees needs to be able to compute the difference between arbitrary trees for completely unrelated sentences. The function δ can be any function as long as it is a metric; that is, it must be (1) non-negative, (2) symmetric, (3) zero only for identical inputs, and (4) it must obey the triangle inequality:

1. $\forall x, y : \delta(x, y) \geq 0$
2. $\forall x, y : \delta(x, y) = \delta(x, y)$
3. $\forall x, y : \delta(x, y) = 0 \Leftrightarrow x = y$
4. $\forall x, y, z : \delta(x, y) + \delta(y, z) \geq \delta(x, z)$

This immediately excludes metrics like PARSEVAL (Black et al., 1991) and Leaf-Ancestor (Sampson and Babarczy, 2003), since they assume that the trees being compared are parses of the same sentence. Instead, we base our work on tree edit distance. The tree edit distance (TED) problem is defined analogously to the more familiar problem of string edit distance: what is the minimum number of edit operations required to transform one tree into the other? See Bille (2005) for a thorough introduction to the tree edit distance problem and other related problems. For this work, we used the algorithm of Zhang and Shasha (1989). Tree edit distance has previously been used in the TEDEVAL software (Tsarfaty et al., 2011; Tsarfaty et al., 2012) for parser evaluation agnostic to both annotation scheme and theoretical framework, but this by itself is still an

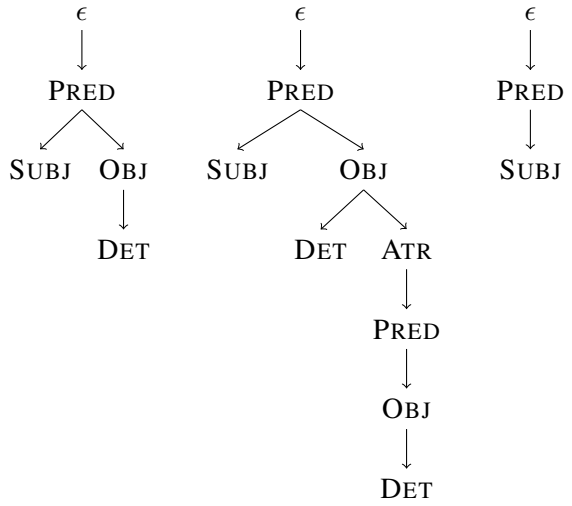


Figure 2: Three trees with distance zero using δ_{diff}

uncorrected accuracy measure and thus unsuitable for our purposes.[3]

When comparing syntactic trees, we only want to compare dependency relations or non-terminal categories. Therefore we remove the leaf nodes in the case of phrase structure trees, and in the case of dependency trees we compare trees whose edges are unlabelled and nodes are labelled with the dependency relation between that word and its head; the root node receives the label ϵ. An example of this latter transformation is shown in Figure 1.

We propose three different distance functions for the agreement computation: the unmodified tree edit distance function, denoted δ_{plain}, a second function $\delta_{diff}(x, y) = \mathrm{TED}(x, y) - \mathrm{abs}(|x| - |y|)$, the edit distance minus the difference in length between the two sentences, and finally $\delta_{norm}(x, y) = {}^{\mathrm{TED}(x,y)}/_{|x|+|y|}$, the edit distance normalised to the range $[0, 1]$.[4]

The plain TED is the simplest in terms of parsimony assumptions, however it may overestimate the difference between sentences, we intuitively find to be syntactically similar. For example the only difference between the two leftmost trees in Figure 2 is a modifier, but δ_{plain} gives them distance 4 and δ_{diff} 0. On the other hand, δ_{diff} might underestimate some distances as well; for example the leftmost and rightmost trees also have distance zero using δ_{diff}, despite our syntactic intuition that the difference between a transitive and an intransitive should be taken account of.

The third distance function, δ_{norm}, takes into account a slightly different concern; namely that when comparing a long sentence and a short sentence, the distance has to be quite large simply to account for the difference in number of nodes, unlike comparing two short or two long sentences. Normalising to the range $[0, 1]$ puts all pairs on an equal footing.

However, we cannot *a priori* say which of the three functions is the optimal choice of distance functions. The different functions have different properties, and different advantages and drawbacks, and the nature of their strengths and weaknesses differ. We will therefore perform a number of synthetic experiments to investigate their properties in a controlled environment, before applying them to real-world data.

3 Synthetic experiments

In the previous section, we proposed three different agreement metrics α_{plain}, α_{diff} and α_{norm}, each involving different trade-offs. Deciding which of these metrics is the best one for our purposes of judging the consistency of syntactic annotation poses a bit of a conundrum. We could at this point apply our metrics to various real corpora and compare the results, but since the consistency of the corpora is unknown, it's impossible to say whether the best metric is the one resulting in the highest scores, the lowest scores or somewhere in the middle. To properly settle this question, we first performed a number of synthetic experiments to gauge how the different metrics respond to disagreement.

The general approach we take is based on that used by Mathet et al. (2012), adapted to dependency trees. An already annotated corpus, in our case 100 randomly selected sentences from the Norwegian Dependency Treebank (Solberg et al., 2014), are taken as correct and then permuted to produce "annotations" of different quality. For dependency trees, the input corpus is permuted as follows:

1. Each token has a probability $p_{relabel}$ of being assigned a different label uniformly at random from the set of labels used in the corpus.

[3] While it is quite different from other parser evaluation schemes, TEDEVAL does not correct for chance agreement and is thus an uncorrected metric. It could of course form the basis for a corrected metric, given a suitable measure of expected agreement.

[4] We can easily show that $|x| + |y|$ is an upper bound on the TED, corresponding to deleting all nodes in the source tree and inserting all the nodes in the target.

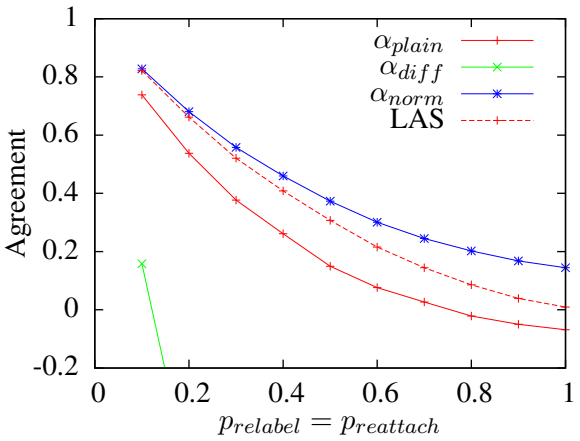


Figure 3: Mean agreement over ten runs

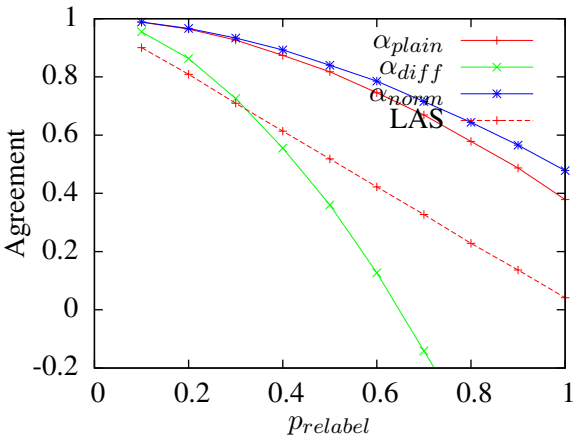


Figure 4: Mean agreement over ten runs, $p_{reattach} = 0$

2. Each token has a probability $p_{reattach}$ of being assigned a new head uniformly at random from the set of tokens not dominated by the token.

The second permutation process is dependent on the order the tokens are processed, and we consider the tokens in the post-order[5] as dictated by the original tree. This way tokens close to the root have a fair chance of having candidate heads if they are selected. A pre-order traversal would result in tokens close to the root having few options, and in particular if the root has a single child, that node has no possible new heads unless one of its children has been assigned the root as its new head first. For example in the trees in figure 2, assigning any other head than the root to the PRED nodes directly dominated by the root will result in invalid (cyclic and unconnected) dependency trees. Traversing the tokens in the linear order dictated by the sentence has similar issues for tokens close to the root and close to the start of the sentence.

For our first set of experiments, we set $p_{relabel} = p_{reattach}$ and evaluated the different agreement metrics for 10 evenly spaced p-values between 0.1 and 1.0. Initial exploration of the data showed that the mean follows the median very closely regardless of metric and perturbation level, and therefore we only report the mean scores across runs in this paper. The results of these experiments are shown in Figure 3, with the labelled attachment score[6] (LAS) for comparison.

The α_{diff} metric is clearly extremely sensitive to noise, with $p = 0.1$ yielding mean $\alpha_{diff} = 15.8\%$, while α_{norm} is more lenient than both LAS and α_{plain}, with mean $\alpha_{norm} = 14.5\%$ at $p = 1$, quite high compared to LAS $= 0.9\%$, $\alpha_{plain} = -6.8\%$ and $\alpha_{diff} = -246\%$. To further study the sensitivity of the metrics to the two kinds of noise, we performed an additional set of experiments, setting one $p = 0$ while varying the other over the same range as in the previous experiment, the results of which are shown in Figures 4 and 5.

The LAS curves are mostly unremarkable, with one exception: Mean LAS at $p_{reattach} = 1$ of Figure 5 is 23.9%, clearly much higher than we would expect if the trees were completely random. In comparison, mean LAS when only labels are perturbed is 4.1%, and since the sample space of trees of size n is clearly much larger than that of relabellings, a uniform random selection of tree would yield a LAS much closer to 0. This shows that our tree shuffling algorithm has a non-uniform distribution over the sample space.

While the behaviour of our alphas and LAS are relatively similar in Figure 3, Figures 4 and 5 show that they do in fact have important differences. Whereas LAS responds linearly to perturbation of both labels and structure, with its parabolic behaviour in Figure 3 being simply the product of these two linear responses, the α metrics respond differently to structural noise and label noise, with label disagreements being penalised less harshly

[5] That is, the child nodes of a node are all processed before the node itself. Nodes on the same level are traversed from left to right.

[6] The *de facto* standard parser evaluation metric in dependency parsing: the percentage of tokens that receive the correct head *and* dependency relation.

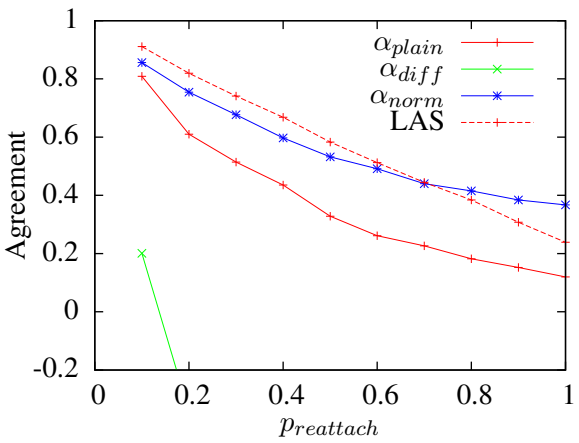


Figure 5: Mean agreement over ten runs, $p_{relabel} = 0$

than structural disagreements.

The reason for the strictness of the α_{diff} metric and the laxity of α_{norm} is the effects the modified distance functions have on the distribution of distances. The δ_{diff} function causes an extreme shift of the distances towards 0; more than 30% of the sentence pairs have distance 0, 1, or 2, which causes D_e^{diff} to be extremely low and thus gives disproportionally large weight to non-zero distances in D_o^{diff}. On the other hand δ_{norm} causes a rightward shift of the distances, which results in a high D_e^{norm} and thus individual disagreements having less weight.

4 Real-world corpora

Synthetic experiments do not always fully reflect real-world behaviour, however. Therefore we will also evaluate our metrics on real-world inter-annotator agreement data sets. In our evaluation, we will contrast labelled accuracy, the standard parser evaluation metric, and our three α metrics. In particular, we are interested in the correlation (or lack thereof) between LAS and the alphas, and whether the results of our synthetic experiments correspond well with the results on real-world IAA sets. Finally, we also evaluate the metric on both dependency and phrase structure data.

4.1 The corpora

We obtained[7] data from four different corpora. Three of the data sets are dependency treebanks (NDT, CDT, PCEDT) and one phrase structure treebank (SSD), and of the dependency treebanks the PCEDT contains semantic dependencies, while the other two have traditional syntactic dependencies. The number of annotators and sizes of the different data sets are summarised in Table 1.

Corpus	Sentences	Tokens
NDT 1[a]	130	1674
NDT 2[a]	110	1594
NDT 3[a]	150	1997
CDT (da)[a]	162	2394
CDT (en)[a]	264	5528
CDT (es)[b]	55	924
CDT (it)[c]	136	3057
PCEDT[d]	3531	61737
SSD[e]	96	1581

[a] 2 annotators
[b] 4 annotators, avg. 2.8 annotators/text (min. 2, max. 4)
[c] 3 annotators, avg. 2.7 annotators/text
[d] 11 annotators, avg. 2.5 annotators/text (min. 2, max. 6)
[e] 3 annotators, avg. 2.9 annotators/sent.

Table 1: Sizes of the different IAA corpora

NDT The Norwegian Dependency Treebank (Solberg et al., 2014) is a dependency treebank constructed at the National Library of Norway. The data studied in this work has previously been used by Skjærholt (2013) to study agreement, but using simple accuracy measures (UAS, LAS) rather than chance-corrected measures. The IAA data set is divided into three parts, corresponding to different parsers used to preprocess the data before annotation; what we term NDT 1 through 3 correspond to what Skjærholt (2013) labels Danish, Swedish and Norwegian, respectively.

CDT The Copenhagen Dependency Treebanks (Buch-Kromann et al., 2009; Buch-Kromann and Korzen, 2010) is a collection of parallel dependency treebanks, containing data from the Danish PAROLE corpus (Keson, 1998b; Keson, 1998a) in the original Danish and translated into English, Italian and Spanish.

PCEDT The Prague Czech-English Dependency Treebank 2.0 Hajič et al. (2012) is a parallel corpus of English and Czech, consisting of English data from the Wall Street Journal Section of the Penn Treebank (Marcus et al., 1993) and

[7] We contacted a number of treebank projects, among them the Penn Treebank and the Prague Dependency Treebank, but not all of them had data available.

Czech translations of the English data. The syntactic annotations are layered and consist of an analytical layer similar to the annotations in most other dependency treebanks, and a more semantic tectogrammatical layer.

Our data set consists of a common set of analytical annotations shared by all the annotators, and the tectogrammatical analyses built on top of this common foundation. A distinguishing feature of the tectogrammatical analyses, vis a vis the other treebanks we are using, is that semantically empty words only take part in the analytical annotation layer and nodes are inserted at the tectogrammatical layer to represent covert elements of the sentence not present in the surface syntax of the analytical layer. Thus, inserting and deleting nodes is a central part of the task of tectogrammatical annotation, unlike the more surface-oriented annotation of our other treebanks, where the tokenisation is fixed before the text is annotated.

SSD The Star-Sem Data is a portion of the dataset released for the *SEM 2012 shared task (Morante and Blanco, 2012), parsed using the LinGO English Resource Grammar (ERG, Flickinger, 2000) and the resulting parse forest disambiguated based on discriminants. The ERG is an HPSG-based grammar, and as such its analyses are attribute-value matrices (AVMs); an AVM is not a tree but a directed acyclic graph however, and for this reason we compute agreement not on the AVM but the so-called *derivation tree*. This tree describes the types of the lexical items in the sentence and the bottom-up ordering of rule applications used to produce the final analysis and can be handled by our procedure like any phrase-structure tree.

4.2 Agreement results

To evaluate our corpora, we compute the three α variants described in the previous two sections, and compare these with labelled accuracy scores.

When there are more than two annotators, we generalise the metric to be the average pairwise LAS for each sentence, weighted by the length of the sentence. Let $\mathrm{LAS}(t_1, t_2)$ be the fraction of tokens with identical head and label in the trees t_1 and t_2; the pairwise labelled accuracy $\mathrm{LAS}_p(X)$ of a set of annotations X as described in section 1.2 is:

$$\mathrm{LAS}_p(X) = \frac{1}{\sum_i |x_{i1}|} \sum \frac{|x_{i1}|\Lambda(X_i)}{|X_i|(|X_i|-1)/2} \quad (3)$$

$$\Lambda(X_i) = \sum_{c=1}^{|C|} \sum_{c'=c+1}^{|C|} \mathrm{LAS}(x_{ic}, x_{ic'})$$

This is equivalent to the traditional metric in the case where there are only two annotators.

As our uncorrected metric for comparing two phrase structure trees we do not use the traditional bracket F_1 as it does not generalise well to more than two annotators, but rather Jaccard similarity. The Jaccard similarity of two sets A and B is the ratio of the size of their intersection to the size of their union: $J(A, B) = |A \cap B| / |A \cup B|$, and we use the Jaccard similarity of the sets of labelled bracketings of two trees as our uncorrected measure. To compute the similarity for a complete set of annotations we use the mean pairwise Jaccard similarity weighted by sentence length; that is, the same procedure as in 3, but using Jaccard similarity rather than LAS.

Since LAS assumes that both of the sentences compared have identical sets of tokens, we had to exclude a number of sentences from the LAS computation in the cases of the English and Italian CDT corpora, and especially the PCEDT. The large number of sentences excluded in the PCEDT is due to the fact that in the tectogrammatical analysis of the PCEDT, inserting and deleting nodes is an important part of the annotation task.

Looking at the results in Table 2, we observe

Corpus	α_{plain}	α_{diff}	α_{norm}	LAS
NDT 1	98.4	93.0	98.8	94.0
NDT 2	98.9	95.0	99.1	94.4
NDT 3	97.9	91.2	98.7	95.3
CDT (da)	95.7	84.7	96.2	90.4
CDT (en)	92.4	70.7	95.0	88.4
CDT (es)	86.6	48.8	85.8	78.9[a]
CDT (it)	84.5	55.7	89.2	81.3[b]
PCEDT	95.9	89.9	96.5	68.0[c]
SSD	99.1	98.6	99.3	87.9[d]

[a] 2 sentences ignored
[b] 15 sentences ignored
[c] 1178 sentences ignored
[d] Mean pairwise Jaccard similarity

Table 2: Agreement scores on real-world corpora

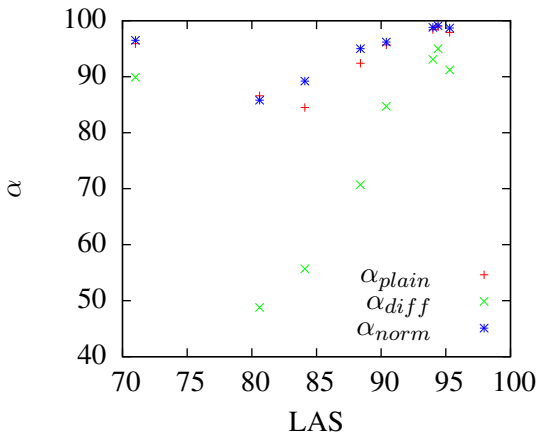


Figure 6: Correlation of LAS with α

two things. Most obvious, is the extremely large gap between the LAS and α metrics for the PCEDT data. However, there is a more subtle point; the orderings of the corpora by the different metrics are not the same. LAS order the corpora NDT 3, 2, 1, CDT da, en, it, es, PCEDT, whereas α_{diff} and α_{norm} gives the order NDT 2, 1, 3, PCEDT, CDT da, en, it, es, and α_{plain} gives the same order as the other alphas but with CDT es and it changing places. Furthermore, as the scatterplot in Figure 6 shows, there is a clear correlation between the α metrics and LAS, if we disregard the PCEDT results.

The reason the PCEDT gets such low LAS is essentially the same as the reason many sentences had to be excluded from the computation in the first place; since inserting and deleting nodes is an integral part of the tectogrammatical annotation task, the assumption implicit in the LAS computation that sentences with the same number of nodes have the same nodes in the same order is obviously false, resulting in a very low LAS.

The corpus that scores the highest for all three metrics is the SSD corpus; the reason for this is uncertain, as our corpora differ along many dimensions, but the fact that the annotation was done by professional linguists who are very familiar with the grammar used to parse the data is likely a contributing factor. The difference between the α metrics and the Jaccard similarity is larger than the difference between α and LAS for our dependency corpora, however the two similarity metrics are not comparable, and it is well known that for phrase structures single disagreements such as a PP-attachment disagreement can result in multiple disagreeing bracketings.

5 Conclusion

The most important conclusion we draw from this work is the most appropriate agreement metric for syntactic annotation. First of all, we disqualify the LAS metric, primarily due to the methodological inadequacies of using an uncorrected measure. While our experiments did not reveal any serious shortcomings (unlike those of Mathet et al., 2012 who in the case of categorisation showed that for large p the uncorrected measure can be *increasing*), the methodological problems of uncorrected metrics makes us wary of LAS as an agreement metric. Next, of the three α metrics, α_{plain} is clearly the best; α_{diff} is extremely sensitive to even moderate amounts of disagreement, while α_{norm} is overly lenient.

Looking solely at Figure 3, one might be led to believe that LAS and α_{plain} are interchangeable, but this is not the case. As shown by Figures 4 and 5, the paraboloid shape of the LAS curve in Figure 3 is simply the combination of the metric's linear responses to both label and structural perturbations. The behaviour of α on the other hand is more complex, with structural noise being penalised harder than perturbations of the labels. Thus, the similarity of LAS and α_{plain} is not at all assured when the amounts of structural and labelling disagreements differ. Additionally, we consider this imbalanced weighting of structural and labelling disagreements a benefit, as structure is the larger part of syntactic annotation compared to the labelling of the dependencies/bracketings. Finally our experiments show that α is a single metric that is applicable to both dependencies and phrase structure trees.

Furthermore, α metrics are far more flexible than simple accuracy metrics. The use of a distance function to define the metric means that more fine-grained distinctions can be made; for example, if the set of labels on the structures is highly structured, partial credit can be given for differing annotations that overlap. For example, if different types of adverbials (temporal, negation, etc.) receive different relations, as is the case in the Swedish Talbanken05 (Nivre et al., 2006) corpus, confusion of different adverbial types can be given less weight than confusion between subject and object. The α-based metrics are also far easier to apply to a more complex annotation task such

as the tectogrammatical annotation of the PCEDT. In this task inserting and deleting nodes is an integral part of the annotation, and if two annotators insert or delete different nodes the all-or-nothing requirement of identical yield of the LAS metric makes it impossible as an evaluation metric in this setting.

5.1 Future work

In future work, we would like to investigate the use of other distance functions, in particular the use of approximate tree edit distance functions such as the pq-gram algorithm (Augsten et al., 2005). For large data sets such as the PCEDT set used in this work, computing α with tree edit distance as the distance measure can take a very long time.[8] This is due to the fact that α requires $O(n^2)$ comparisons to be made, each of which is $O(n^2)$ using our current approach. The problem of directed graph edit distance is NP-hard, which means that to apply our method to HPSG analyses directly approximate algorithms are a requirement.

Another avenue for future work is improved synthetic experiments. As we saw, our implementation of tree perturbations was biased towards trees similar in shape to the source tree, and an improved permutation algorithm may reveal interesting edge-case behaviour in the metrics. A method for perturbing phrase structure trees would also be interesting, as this would allow us to repeat the synthetic experiments performed here using phrase structure corpora to compare the behaviour of the metrics on the two types of corpus.

Finally, annotator modelling techniques like that presented in Passonneau and Carpenter (2013) has obvious advantages over agreement coefficients such as α. These techniques are interpreted more easily than agreement coefficients, and they allow us to assess the quality of individual annotators, a crucial property in crowd-sourcing settings and something that's impossible using agreement coefficients.

Acknowledgements

I would like to thank Jan Štěpánek at Charles University for data from the PCEDT and help with the conversion process, the CDT project for publishing their agreement data, Per Erik Solberg at the Norwegian National Library for data from the NDT, and Emily Bender at the University of Washington for the SSD data.

[8]The Python implementation used in this work, using NumPy and the PyPy compiler, took seven and a half hours compute a single α for the PCEDT data set on an Intel Core i7 2.9 GHz computer. The program is single-threaded.

References

Ron Artstein and Massimo Poesio. 2008. Inter-Coder Agreement for Computational Linguistics. *Computational Linguistics*, 34(4):555–596.

Nikolaus Augsten, Böhlen Michael, and Johann Gamper. 2005. Approximate Matching of Hierarchical Data Using pq-Grams. In *Proceedings of the 31st international conference on Very large data bases*, pages 301–312, Trondheim. VLDB Endowment.

E. M. Bennett, R. Alpert, and A. C. Goldstein. 1954. Communications Through Limited-Response Questioning. *Public Opinion Quarterly*, 18(3):303–308.

Riyaz Ahmad Bhat and Dipti Misri Sharma. 2012. A Dependency Treebank of Urdu and its Evaluation. In *Proceedings of the Sixth Linguistic Annotation Workshop*, pages 157–165, Jeju. Association for Computational Linguistics.

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

Ezra Black, Steven Abney, Dan Flickinger, Claudia Gdaniec, Ralph Grishman, Philip Harrison, Donald Hindle, Robert Ingria, Frederick Jelinek, Judith Klavans, Mark Liberman, Mitchell P. Marcus, Salim Roukos, Beatrice Santorini, and Tomek Strzalkowski. 1991. A Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars. In *Proceedings of the workshop on Speech and Natural Language*, pages 306–311, Pacific Grove, USA.

Sabine Brants and Silvia Hansen. 2002. Developments in the TIGER Annotation Scheme and their Realization in the Corpus. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, pages 1643–1649.

Thorsten Brants. 2000. Inter-Annotator Agreement for a German Newspaper Corpus. In *Proceedings of the Second International Conference on Language Resources and Evaluation*.

Matthias Buch-Kromann and Iørn Korzen. 2010. The unified annotation of syntax and discourse in the Copenhagen Dependency Treebanks. In *Proceedings of the Fourth Linguistic Annotation Workshop*, pages 127–131, Uppsala. Association for Computational Linguistics.

Matthias Buch-Kromann, Iørn Korzen, and Henrik Høeg Müller. 2009. Uncovering the 'lost' structure of translations with parallel treebanks. In Fabio Alves, Susanne Göpferich, and Inger Mees, editors,

Methodology, Technology and Innovation in Translation Process Research, pages 199–224. Samfundslitteratur, Frederiksberg.

Jean Carletta. 1996. Assessing Agreement on Classification Tasks: The Kappa Statistic. *Computational Linguistics*, 22(2):249–254.

Montserrat Civit, Alicia Ageno, Borja Navarro, Núria Bufí, and M. Antònia Martí. 2003. Qualitative and Quantitative Analysis of Annotators' Agreement in the Development of. In Joakim Nivre and Erhard Hinrichs, editors, *Proceedings of the Second Workshop on Treebanks and Linguistic Theories*, pages 21–32, Växjö. Växjö University Press.

Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37–46, April.

Sérgio Ricardo de Castro. 2011. *Developing reliability metrics and validation tools for datasets with deep linguistic information*. Master's thesis, Universidade de Lisboa.

Jason M. Eisner. 1996. Three New Probabilistic Models for Dependency Parsing: An Exploration. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 340–345, Stroudsburg. Association for Computational Linguistics.

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6(1):15–28, March.

Chris Fournier. 2013. Evaluating Text Segmentation using Boundary Edit Distance. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1702–1712, Sofia. Association for Computational Linguistics.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. 2012. Prague Czech-English Dependency Treebank 2.0.

Jan Hajič. 2004. Complex Corpus Annotation: The Prague Dependency Treebank. Jazykovedný ústav Ľ. Štúra, SAV.

Britt Keson. 1998a. The Danish Morphosyntactically Tagged PAROLE Corpus. Technical report, Danish Society for Literature and Language, Copenhagen.

Britt Keson. 1998b. Vejledning til det danske morfosyntaktisk taggede PAROLE-korpus. Technical report, Danish Society for Literature and Language, Copenhagen.

Klaus Krippendorff. 1970. Estimating the Reliability, Systematic Error and Random Error of Interval Data. *Educational and Psychological Measurement*, 30(1):61–70, April.

Klaus Krippendorff. 2004. *Content Analysis: An introduction to its methodology*. Sage Publications, Thousand Oaks, 2nd edition.

Mohamed Maamouri, Ann Bies, and Seth Kulick. 2008. Enhancing the Arabic Treebank : A Collaborative Effort toward New Annotation Guidelines. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation*, pages 3192–3196. European Language Resources Association.

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

Yann Mathet, Antoine Widlöcher, Karën Fort, Claire François, Olivier Galibert, Cyril Grouin, Juliette Kahn, Sophie Rosset, and Pierre Zweigenbaum. 2012. Manual Corpus Annotation : Giving Meaning to the Evaluation Metrics. In *Proceedings of COLING 2012*, pages 809–818, Mumbai.

Marie Mikulová and Jan Štěpánek. 2010. Ways of Evaluation of the Annotators in Building the Prague Czech-English Dependency Treebank. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation*, pages 1836–1839, Valletta. European Language Resources Association.

Roser Morante and Eduardo Blanco. 2012. *SEM 2012 Shared Task : Resolving the Scope and Focus of Negation. In *The First Joint Conference on Lexical and Computational Semantics*, pages 265–274, Montreal. Association for Computational Linguistics.

Joakim Nivre, Jens Nilsson, and Johan Hall. 2006. Talbanken05 : A Swedish Treebank with Phrase Structure and Dependency Annotation. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*.

Rebecca J. Passonneau and Bob Carpenter. 2013. The Benefits of a Model of Annotation. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 187–195, Sofia. Association for Computational Linguistics.

Rebecca J. Passonneau. 2006. Measuring Agreement on Set-valued Items (MASI) for Semantic and Pragmatic Annotation. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, pages 831–836.

Marwa Ragheb and Markus Dickinson. 2013. Inter-annotator Agreement for Dependency Annotation of Learner Language. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 169–179, Atlanta. Association for Computational Linguistics.

Geoffrey Sampson and Anna Babarczy. 2003. A test of the leaf-ancestor metric for parse accuracy. *Natural Language Engineering*, 9(4):365–380, December.

William A. Scott. 1955. Reliability of Content Analysis: The Case of Nominal Scale Coding. *Public Opinion Quarterly*, 19(3):321–325, January.

Arne Skjærholt. 2013. Influence of preprocessing on dependency syntax annotation: speed and agreement. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 28–32, Sofia. Association for Computational Linguistics.

Per Erik Solberg, Arne Skjærholt, Lilja Øvrelid, Kristin Hagen, and Janne Bondi Johannesen. 2014. The Norwegian Dependency Treebank. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, Reykjavik. European Language Resources Association.

Reut Tsarfaty, Joakim Nivre, and Evelina Andersson. 2011. Evaluating Dependency Parsing: Robust and Heuristics-Free Cross-Annotation Evaluation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 385–396, Edinburgh. Association for Computational Linguistics.

Reut Tsarfaty, Joakim Nivre, and Evelina Andersson. 2012. Cross-Framework Evaluation for Statistical Parsing. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 44–54, Avignon. Association for Computational Linguistics.

Kaizhong Zhang and Dennis Shasha. 1989. Simple Fast Algorithms for the Editing Distance between Trees and Related Problems. *SIAM Journal on Computing*, 18(6):1245–1262, December.

Two Is Bigger (and Better) Than One: the Wikipedia Bitaxonomy Project

Tiziano Flati, Daniele Vannella, Tommaso Pasini and **Roberto Navigli**
Dipartimento di Informatica
Sapienza Università di Roma
{flati,vannella,navigli}@di.uniroma1.it
p.tommaso@gmail.com

Abstract

We present WiBi, an approach to the automatic creation of a bitaxonomy for Wikipedia, that is, an integrated taxonomy of Wikipage pages and categories. We leverage the information available in either one of the taxonomies to reinforce the creation of the other taxonomy. Our experiments show higher quality and coverage than state-of-the-art resources like DBpedia, YAGO, MENTA, WikiNet and WikiTaxonomy. WiBi is available at http://wibitaxonomy.org.

1 Introduction

Knowledge has unquestionably become a key component of current intelligent systems in many fields of Artificial Intelligence. The creation and use of machine-readable knowledge has not only entailed researchers (Mitchell, 2005; Mirkin et al., 2009; Poon et al., 2010) developing huge, broad-coverage knowledge bases (Hovy et al., 2013; Suchanek and Weikum, 2013), but it has also hit big industry players such as Google (Singhal, 2012) and IBM (Ferrucci, 2012), which are moving fast towards large-scale knowledge-oriented systems.

The creation of very large knowledge bases has been made possible by the availability of collaboratively-curated online resources such as Wikipedia and Wiktionary. These resources are increasingly becoming enriched with new content in many languages and, although they are only partially structured, they provide a great deal of valuable knowledge which can be harvested and transformed into structured form (Medelyan et al., 2009; Hovy et al., 2013). Prominent examples include DBpedia (Bizer et al., 2009), BabelNet (Navigli and Ponzetto, 2012), YAGO (Hoffart et al., 2013) and WikiNet (Nastase and Strube, 2013). The types of semantic relation in these resources range from domain-specific, as in Freebase (Bollacker et al., 2008), to unspecified relations, as in BabelNet. However, unlike the case with smaller manually-curated resources such as WordNet (Fellbaum, 1998), in many large automatically-created resources the taxonomical information is either missing, mixed across resources, e.g., linking Wikipedia categories to WordNet synsets as in YAGO, or coarse-grained, as in DBpedia whose hypernyms link to a small upper taxonomy.

Current approaches in the literature have mostly focused on the extraction of taxonomies from the network of Wikipedia categories. WikiTaxonomy (Ponzetto and Strube, 2007), the first approach of this kind, is based on the use of heuristics to determine whether is-a relations hold between a category and its subcategories. Subsequent approaches have also exploited heuristics, but have extended them to any kind of semantic relation expressed in the category names (Nastase and Strube, 2013). But while the aforementioned attempts provide structure for categories that supply meta-information for Wikipedia pages, surprisingly little attention has been paid to the acquisition of a full-fledged taxonomy for Wikipedia pages themselves. For instance, Ruiz-Casado et al. (2005) provide a general vector-based method which, however, is incapable of linking pages which do not have a WordNet counterpart. Higher coverage is provided by de Melo and Weikum (2010) thanks to the use of a set of effective heuristics, however, the approach also draws on WordNet and sense frequency information.

In this paper we address the task of taxonomizing Wikipedia in a way that is fully independent of other existing resources such as WordNet. We present WiBi, a novel approach to the creation of a Wikipedia bitaxonomy, that is, a taxonomy of Wikipedia pages aligned to a taxonomy of categories. At the core of our approach lies the idea that the information at the page and category

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 945–955,
Baltimore, Maryland, USA, June 23-25 2014.

level are mutually beneficial for inducing a wide-coverage and fine-grained integrated taxonomy.

2 WiBi: A Wikipedia Bitaxonomy

We induce a Wikipedia bitaxonomy, i.e., a taxonomy of pages and categories, in 3 phases:

1. **Creation of the initial page taxonomy:** we first create a taxonomy for the Wikipedia pages by parsing textual definitions, extracting the hypernym(s) and disambiguating them according to the page inventory.
2. **Creation of the bitaxonomy:** we leverage the hypernyms in the page taxonomy, together with their links to the corresponding categories, so as to induce a taxonomy over Wikipedia categories in an iterative way. At each iteration, the links in the page taxonomy are used to identify category hypernyms and, conversely, the new category hypernyms are used to identify more page hypernyms.
3. **Refinement of the category taxonomy:** finally we employ structural heuristics to overcome inherent problems affecting categories.

The output of our three-phase approach is a bitaxonomy of millions of pages and hundreds of thousands of categories for the English Wikipedia.

3 Phase 1: Inducing the Page Taxonomy

The goal of the first phase is to induce a taxonomy of Wikipedia pages. Let P be the set of all the pages and let $T_P = (P, E)$ be the page taxonomy directed graph whose nodes are pages and whose edge set E is initially empty ($E := \emptyset$). For each $p \in P$ our aim is to identify the most suitable generalization $p_h \in P$ so that we can create the edge (p, p_h) and add it to E. For instance, given the page APPLE, which represents the fruit meaning of *apple*, we want to determine that its hypernym is FRUIT and add the hypernym edge connecting the two pages (i.e., $E := E \cup \{(\text{APPLE}, \text{FRUIT})\}$). To do this, we perform a syntactic step, in which the hypernyms are extracted from the page's textual definition, and a semantic step, in which the extracted hypernyms are disambiguated according to the Wikipedia inventory.

3.1 Syntactic step: hypernym extraction

In the syntactic step, for each page $p \in P$, we extract zero, one or more hypernym lemmas, that is, we output potentially ambiguous hypernyms for the page. The first assumption, which follows

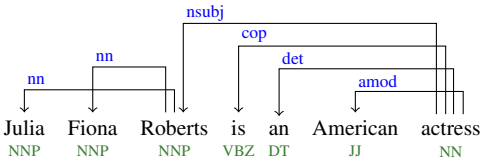


Figure 1: A dependency tree example with copula.

the Wikipedia guidelines and is validated in the literature (Navigli and Velardi, 2010; Navigli and Ponzetto, 2012), is that the first sentence of each Wikipedia page p provides a textual definition for the concept represented by p. The second assumption we build upon is the idea that a lexical taxonomy can be obtained by extracting hypernyms from textual definitions. This idea dates back to the early 1970s (Calzolari et al., 1973), with later developments in the 1980s (Amsler, 1981; Calzolari, 1982) and the 1990s (Ide and Véronis, 1993).

To extract hypernym lemmas, we draw on the notion of copula, that is, the relation between the complement of a copular verb and the copular verb itself. Therefore, we apply the Stanford parser (Klein and Manning, 2003) to the definition of a page in order to extract all the dependency relations of the sentence. For example, given the definition of the page JULIA ROBERTS, i.e., "Julia Fiona Roberts is an American actress.", the Stanford parser outputs the set of dependencies shown in Figure 1. The noun involved in the copula relation is *actress* and thus it is taken as the page's hypernym lemma. However, the extracted hypernym is sometimes overgeneral (*one*, *kind*, *type*, etc.). For instance, given the definition of the page APOLLO, "Apollo is one of the most important and complex of the Olympian deities in ancient Greek and Roman religion [...].", the only copula relation extracted is between *is* and *one*. To cope with this problem we use a list of stopwords.[1]When such a term is extracted as hypernym, we replace it with the rightmost noun of the first following noun sequence (e.g., *deity* in the above example). If the resulting lemma is again a stopword we repeat the procedure, until a valid hypernym or no appropriate hypernym can be found. Finally, to capture multiple hypernyms, we iteratively follow the *conj_and* and *conj_or* relations starting from the initially extracted hypernym. For example, consider the definition of ARISTOTLE: "Aristotle was a Greek philosopher and polymath, a student of Plato and teacher of Alexander the Great." Initially, the *philosopher* hypernym is selected thanks to the copula relation, then, fol-

[1]E.g., *species*, *genus*, *one*, etc. Full list available online.

lowing the conjunction relations, also *polymath*, *student* and *teacher* are extracted as hypernyms. While more sophisticated approaches like Word-Class Lattices could be applied (Navigli and Velardi, 2010), we found that, in practice, our hypernym extraction approach provides higher coverage, which is critical in our case.

3.2 Semantic step: hypernym disambiguation

Since our aim is to connect pairs of pages via hypernym relations, our second step consists of disambiguating the obtained hypernym lemmas of page p by associating the most suitable page with each hypernym. Following previous work (Ruiz-Casado et al., 2005; Navigli and Ponzetto, 2012), as the inventory for a given lemma we consider the set of pages whose main title is the lemma itself, except for the sense specification in parenthesis. For instance, given *fruit* as the hypernym for APPLE we would like to link APPLE to FRUIT as opposed to, e.g., FRUIT (BAND) or FRUIT (ALBUM).

3.2.1 Hypernym linkers

To disambiguate hypernym lemmas, we exploit the structural features of Wikipedia through a pipeline of hypernym linkers $\mathcal{L} = \{L_i\}$, applied in cascade order (cf. Section 3.3.1). We start with the set of page-hypernym pairs $H = \{(p, h)\}$ as obtained from the syntactic step. The successful application of a linker to a pair $(p, h) \in H$ yields a page p_h as the most suitable sense of h, resulting in setting $isa(p, h) = p_h$. At step i, the i-th linker $L_i \in \mathcal{L}$ is applied to H and all the hypernyms which the linker could disambiguate are removed from H. This prevents lower-precision linkers from overriding decisions taken by more accurate ones.

We now describe the hypernym linkers. In what follows we denote with $p \xrightarrow{h} p_h$ the fact that the definition of a Wikipedia page p contains an occurrence of h linked to page p_h. Note that p_h is not necessarily a sense of h.

Crowdsourced linker If $p \xrightarrow{h} p_h$, i.e., the hypernym h is found to have been manually linked to p_h in p by Wikipedians, we assign $isa(p, h) = p_h$. For example, because *capital* was linked in the BRUSSELS page definition to CAPITAL CITY, we set isa(BRUSSELS, *capital*) = CAPITAL CITY.

Category linker Given the set $W \subset P$ of Wikipedia pages which have at least one category in common with p, we select the majority sense of h, if there is one, as hyperlinked across all the definitions of pages in W:

$$isa(p, h) = \arg\max_{p_h} \sum_{p' \in W} 1(p' \xrightarrow{h} p_h)$$

where $1(p' \xrightarrow{h} p_h)$ is the characteristic function which equals 1 if h is linked to p_h in page p', 0 otherwise. For example, the linker sets isa(EGGPLANT, *plant*) = PLANT because most of the pages associated with TROPICAL FRUIT, a category of EGGPLANT, contain in their definitions the term *plant* linked to the PLANT page.

Multiword linker If $p \xrightarrow{m} p_h$ and m is a multiword expression containing the lemma h as one of its words, set $isa(p, h) = p_h$. For example, we set isa(PROTEIN, *compound*) = CHEMICAL COMPOUND, as *chemical compound* is linked to CHEMICAL COMPOUND in the definition of PROTEIN.

Monosemous linker If h is monosemous in Wikipedia (i.e., there is only a single page p_h for that lemma), link it to its only sense by setting $isa(p, h) = p_h$. For example, we extract the hypernym *businessperson* from the definition of MERCHANT and, as it is unambiguous, we link it to BUSINESSPERSON.

Distributional linker Finally, we provide a distributional approach to hypernym disambiguation. We represent the textual definition of page p as a distributional vector $\vec{v}_p$ whose components are all the English lemmas in Wikipedia. The value of each component is the occurrence count of the corresponding content word in the definition of p.

The goal of this approach is to find the best link for hypernym h of p among the pages h is linked to, across the whole set of definitions in Wikipedia. Formally, for each p_h such that h is linked to p_h in some definition, we define the set of pages $P(p_h)$ whose definitions contain a link to p_h, i.e., $P(p_h) = \{p' \in P | p' \xrightarrow{h} p_h\}$. We then build a distributional vector $\vec{v}_{p'}$ for each $p' \in P(p_h)$ as explained above and create an aggregate vector $\vec{v}_{p_h} = \sum_{p'} \vec{v}_{p'}$. Finally, we determine the similarity of p to each p_h by calculating the dot product between the two vectors $sim(p, p_h) = \vec{v}_p \cdot \vec{v}_{p_h}$. If $sim(p, p_h) > 0$ for any p_h we perform the following association:

$$isa(p, h) = \arg\max_{p_h} sim(p, p_h)$$

For example, thanks to this linker we set isa(VACUUM CLEANER, *device*) = MACHINE.

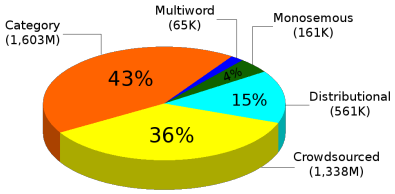


Figure 2: Distribution of linked hypernyms.

3.3 Page Taxonomy Evaluation

Statistics We applied the above linkers to the October 2012 English Wikipedia dump. Out of the 3,829,058 total pages, 4,270,232 hypernym lemmas were extracted in the syntactic step for 3,697,113 pages (covering more than 96% of the total). Due to illformed definitions, though, it was not always possible to extract the hypernym lemma: for example, 6 APRIL 2010 BAGHDAD BOMBINGS is defined as "A series of bomb explosions destroyed several buildings in Baghdad", which only implicitly provides the hypernym.

The semantic step disambiguated 3,718,612 hypernyms for 3,294,562 Wikipedia pages, i.e., covering more than 86% of the English pages with at least one disambiguated hypernym. Figure 2 plots the number and distribution of hypernyms disambiguated by our hypernym linkers.

Taxonomy quality To evaluate the quality of our page taxonomy we randomly sampled 1,000 Wikipedia pages. For each page we provided: i) a list of suitable hypernym lemmas for the page, mainly selected from its definition; ii) for each lemma the correct hypernym page(s). We calculated precision as the average ratio of correct hypernym lemmas (senses) to the total number of lemmas (senses) returned for all the pages in the dataset, recall as the number of correct lemmas (senses) over the total number of lemmas (senses) in the dataset, and coverage as the fraction of pages for which at least one lemma (sense) was returned, independently of its correctness. Results, both at lemma- and sense-level, are reported in Table 1. Not only does our taxonomy show high precision and recall in extracting ambiguous hypernyms, it also disambiguates more than 3/4 of the hypernyms with high precision.

3.3.1 Hypernym linker order

The optimal order of application of the above linkers is the same as that presented in Section 3.2.1. It was established by selecting the combination, among all possible permutations, which maximized precision on a tuning set of 100 randomly sampled pages, disjoint from our page dataset.

	Prec.	Rec.	Cov.
Lemma	94.83	90.20	98.50
Sense	82.77	75.10	89.20

Table 1: Page taxonomy performance.

4 Phase 2: Inducing the Bitaxonomy

The page taxonomy built in Section 3 will serve as a stable, pivotal input to the second phase, the aim of which is to build our bitaxonomy, that is, a taxonomy of pages and categories. Our key idea is that the generalization-specialization information available in each of the two taxonomies is mutually beneficial. We implement this idea by exploiting one taxonomy to update the other, and vice versa, in an iterative way, until a fixed point is reached. The final output of this phase is, on the one hand, a page taxonomy augmented with additional hypernymy relations and, on the other hand, a category taxonomy which is built from scratch.

4.1 Initialization

Our bitaxonomy $B = \{T_P, T_C\}$ is a pair consisting of the page taxonomy $T_P = (P, E)$, as obtained in Section 3, and the category taxonomy $T_C = (C, \emptyset)$, which initially contains all the categories as nodes but does not include any hypernym edge between category nodes. In the following we describe the core algorithm of our approach, which iteratively and mutually populates and refines the edge sets $E(T_P)$ and $E(T_C)$.

4.2 The Bitaxonomy Algorithm

Preliminaries Before proceeding, we define some basic concepts that will turn out to be useful for presenting our algorithm. We denote by $super_T(t)$ the set of all ancestors of a node t in the taxonomy T (be it T_P or T_C). We further define a verification function $t \rightsquigarrow_T t'$ which, in the case of T_C, returns true if there is a path in the Wikipedia category network between t and t', false otherwise, and, in the case of T_P, returns true if t' is a sense, i.e., a page, of a hypernym h of t (that is, $(t, h) \in H$, cf. Section 3.2.1). For instance, SPORTSMEN $\rightsquigarrow_{T_C}$ MEN BY OCCUPATION holds for categories because the former is a sub-category of the latter in Wikipedia, and RADIOHEAD $\rightsquigarrow_{T_P}$ BAND (MUSIC) for pages, because *band* is a hypernym extracted from the textual definition of RADIOHEAD and BAND (MUSIC) is a sense of *band* in Wikipedia. Note that, while the *super* function returns information that we have already learned, i.e., it is in T_P and T_C, the $\rightsquigarrow$ operator

holds just for candidate is-a relations, as it uses knowledge from Wikipedia itself which is potentially incorrect. For instance, SPORTSMEN $\rightsquigarrow_{T_C}$ MEN'S SPORTS in the Wikipedia category network, and RADIOHEAD $\rightsquigarrow_{T_P}$ BAND (RADIO) between the two Wikipedia pages, both hold according to our definition of $\rightsquigarrow$, while connecting the wrong hypernym candidates. At the core of our algorithm, explained below, is the mutual leveraging of the *super* function from one of the two taxonomies (pages or categories) to decide about which candidates (for which a $\rightsquigarrow$ relation holds) in the other taxonomy are real hypernyms.

Finally, we define the projection operator π, such that $\pi(c)$, $c \in C$, is the set of pages categorized with c, and $\pi(p)$, $p \in P$, is the set of categories associated with page p in Wikipedia. For instance, the pages which belong to the category OLYMPIC SPORTS are given by π(OLYMPIC SPORTS) = {BASEBALL, BOXING, ..., TRIATHLON}. Vice versa, π(TRIATHLON) = {MULTISPORTS, OLYMPIC SPORTS, ..., OPEN WATER SWIMMING}. The projection operator π enables us to jump from one taxonomy to the other and expresses the mutual membership relation between pages and categories.

Algorithm We now describe in detail the bitaxonomy algorithm, whose pseudocode is given in Algorithm 1. The algorithm takes as input the two taxonomies, initialized as described in Section 4.1. Starting from the category taxonomy (line 1), the algorithm updates the two taxonomies in turn, until convergence is reached, i.e., no more edges can be added to any side of the bitaxonomy. Let T be the current taxonomy considered at a given moment and T' its dual taxonomy. The algorithm proceeds by selecting a node $t \in V(T)$ for which no hypernym edge (t, t_h) could be found up until that moment (line 3), and then tries to infer such a relation by drawing on the dual taxonomy T' (lines 5-12). This is the core of the bitaxonomy algorithm, in which hypernymy knowledge is transferred from one taxonomy to the other. By applying the projection operator π to t, the algorithm considers those nodes t' aligned to t in the dual taxonomy (line 5) and obtains their hypernyms t'_h using the $super_{T'}$ function (line 6). The nodes reached in T' act as a clue for discovering the suitable hypernyms for the starting node $t \in V(T)$. To perform the discovery, the algorithm projects each such hypernym node $t'_h \in S$ and increments the count of each projection $t_h \in \pi(t'_h)$ (line 8). Finally, the node $\hat{t}_h \in V(T)$ with maximum count, and such that $t \rightsquigarrow_T \hat{t}_h$ holds, if one exists, is promoted as hypernym of t and a new hypernym edge $(t, \hat{t}_h)$ is added to $E(T)$ (line 12). Finally, the role of the two taxonomies is swapped and the process is repeated until no more change is possible.

Algorithm 1 The Bitaxonomy Algorithm

Input: T_P, T_C
1: $T := T_C, T' := T_P$
2: **repeat**
3: **for all** $t \in V(T)$ s.t. $\nexists(t, t_h) \in E(T)$ **do**
4: **reset** *count*
5: **for all** $t' \in \pi(t)$ **do**
6: $S := super_{T'}(t')$
7: **for all** $t'_h \in S$ **do**
8: **for all** $t_h \in \pi(t'_h)$ **do** $count(t_h)$++ **end for**
9: **end for**
10: **end for**
11: $\hat{t}_h := \arg\max_{t_h:\, t \rightsquigarrow_T t_h} count(t_h)$
12: **if** $count(\hat{t}_h) > 0$ **then** $E(T) := E(T) \cup \{(t, \hat{t}_h)\}$
13: **end for**
14: **swap** T and T'
15: **until** convergence
16: **return** $\{T, T'\}$

Example Let us illustrate the algorithm by way of an example. Assume we are in the first iteration ($T = T_C$) and consider the Wikipedia category t = OLYMPICS (line 3) and its super-categories {MULTI-SPORT EVENTS, SPORT AND POLITICS, INTERNATIONAL SPORTS COMPETITIONS}. This category has 27 pages associated with it (line 5), 23 of which provide a hypernym page in T_P (line 6): e.g., PARALYMPIC GAMES, associated with the category OLYMPICS, is a MULTI-SPORT EVENT and is therefore contained in S. By considering and counting the categories of each page in S (lines 7-8), we end up counting the category MULTI-SPORT EVENTS four times and other categories, such as AWARDS and SWIMSUITS, once. As MULTI-SPORT EVENTS has the highest count and is connected to OLYMPICS by a path in the Wikipedia category network (line 11), the hypernym edge (OLYMPICS, MULTI-SPORT EVENTS) is added to T_C (line 12).

5 Phase 3: Category taxonomy refinement

As the final phase, we refine and enrich the category taxonomy. The goal of this phase is to provide broader coverage to the category taxonomy T_C created as explained in Section 4. We apply three enrichment heuristics which add hypernyms to those categories c for which no hypernym could be found in phase 2, i.e., $\nexists c'$ s.t. $(c, c') \in E(T_C)$.

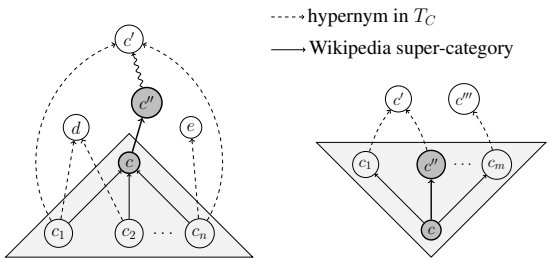
Single super-category As a first structural refinement, we automatically link an uncovered category c to c' if c' is the only direct super-category of c in Wikipedia.

Sub-categories We increase the hypernym coverage by exploiting the sub-categories of each uncovered category c (see Figure 3a). In detail, for each uncovered category c we consider the set $sub(c)$ of all the Wikipedia subcategories of c (nodes $c_1, c_2, \ldots, c_n$ in Figure 3a) and then let each category vote, according to its direct hypernym categories in T_C (the vote is as in Algorithm 1). Then we proceed in decreasing order of vote and select the highest-ranking category c' which is connected to c via a path in T_C, i.e., $c \rightsquigarrow_{T_C} c'$. We then pick up the direct ancestor c'' of c which lies in the path from c to c' and add the hypernym edge (c, c'') to $E(T_C)$. For example, consider the category FRENCH TELEVISION PEOPLE; since this category has no associated pages, in phase 2 no hypernym could be found. However, by applying the sub-categories heuristic, we discover that TELEVISION PEOPLE BY COUNTRY is the hypernym most voted by our target category's descendants, such as FRENCH TELEVISION ACTORS and FRENCH TELEVISION DIRECTORS. Since TELEVISION PEOPLE BY COUNTRY is at distance 1 in the Wikipedia category network from FRENCH TELEVISION PEOPLE, we add (FRENCH TELEVISION PEOPLE, TELEVISION PEOPLE BY COUNTRY) to $E(T_C)$.

Super-categories We then apply a similar heuristic involving super-categories (see Figure 3b). Given an uncovered category c, we consider its direct Wikipedia super-categories and let them vote, according to their hypernym categories in T_C. Then we proceed in decreasing order of vote and select the highest-ranking category c' which is connected to c in T_C, i.e., $c \rightsquigarrow_{T_C} c'$. We then pick up the direct ancestor c'' of c which lies in the path from c to c' and add the edge (c, c'') to $E(T_C)$.

5.1 Bitaxonomy Evaluation

Category taxonomy statistics We applied phases 2 and 3 to the output of phase 1, which was evaluated in Section 3.3. In Figure 4a we show the increase in category coverage at each iteration throughout the execution of the two phases (1SUP, SUB and SUPER correspond to the three above heuristics of phase 3). The final outcome is a category taxonomy which includes 594,917 hypernymy links between categories,



(a) Sub categ. heuristic. (b) Super categ. heuristic.

Figure 3: Heuristic patterns for the coverage refinement of the category taxonomy.

covering more than 96% of the 618,641 categories in the October 2012 English Wikipedia dump. The graph shows the steepest slope in the first iterations of phase 2, which converges around 400k categories at iteration 30, and a significant boost due to phase 3 producing another 175k hypernymy edges, with the super-category heuristic contributing most. 78.90% of the nodes in T_C belong to the same connected component. The average height of the biggest component of T_C is 23.26 edges and the maximum height is 49. We note that the average height of T_C is much greater than that of T_P, which reflects the category taxonomy distinguishing between very subtle classes, such as ALBUMS BY ARTISTS, ALBUMS BY RECORDING LOCATION, etc.

Category taxonomy quality To estimate the quality of the category taxonomy, we randomly sampled 1,000 categories and, for each of them, we manually associated the super-categories which were deemed to be appropriate hypernyms. Figure 4b shows the performance trend as the algorithm iteratively covers more and more categories. Phase 2 is particularly robust across iterations, as it leads to increased recall while retaining very high precision. As regards phase 3, the super-categories heuristic leads to only a slight precision decrease, while improving recall considerably. Overall, the final taxonomy T_C achieves 85.80% precision, 83.40% recall and 97.20% coverage on our dataset.

Page taxonomy improvement As a result of phase 2, 141,105 additional hypernymy links were also added to the page taxonomy, resulting in an overall 82.99% precision, 77.90% recall and 92.10% coverage, with a non-negligible 3% boost from phase 1 to phase 2 in terms of recall and coverage on our Wikipedia page dataset.

We also calculated some statistics for the resulting taxonomy obtained by aggregating the 3.8M

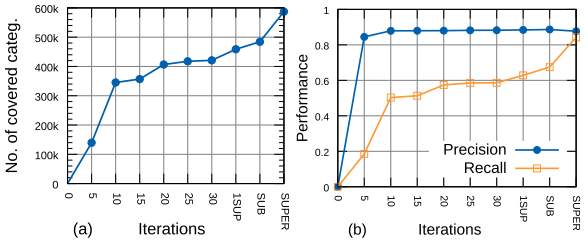


Figure 4: Category taxonomy evaluation.

hypernym links in a single directed graph. Overall, 99% of nodes belong to the same connected component, with a maximum height of 29 and an average height on the biggest component of 6.98.

6 Related Work

Although the extraction of taxonomies from machine-readable dictionaries was already being studied in the early 1970s (Calzolari et al., 1973), pioneering work on large amounts of data only appeared in the 1990s (Hearst, 1992; Ide and Véronis, 1993). Approaches based on handcrafted patterns and pattern matching techniques have been developed to provide a supertype for the extracted terms (Etzioni et al., 2004; Blohm, 2007; Kozareva and Hovy, 2010; Navigli and Velardi, 2010; Velardi et al., 2013, inter alia). However, these methods do not link terms to existing knowledge resources such as WordNet, whereas those that explicitly link do so by adding new leaves to the existing taxonomy instead of acquiring wide-coverage taxonomies from scratch (Pantel and Ravichandran, 2004; Snow et al., 2006).

The recent upsurge of interest in collaborative knowledge curation has enabled several approaches to large-scale taxonomy acquisition (Hovy et al., 2013). Most approaches initially focused on the Wikipedia category network, an entangled set of generalization-containment relations between Wikipedia categories, to extract the hypernymy taxonomy as a subset of the network. The first approach of this kind was WikiTaxonomy (Ponzetto and Strube, 2007; Ponzetto and Strube, 2011), based on simple, yet effective lightweight heuristics, totaling more than 100k is-a relations. Other approaches, such as YAGO (Suchanek et al., 2008; Hoffart et al., 2013), yield a taxonomical backbone by linking Wikipedia categories to WordNet. However, the categories are linked to the first, i.e., most frequent, sense of the category head in WordNet, involving only leaf categories in the linking.

Interest in taxonomizing Wikipedia pages, instead, developed with DBpedia (Auer et al., 2007), which pioneered the current stream of work aimed at extracting semi-structured information from Wikipedia templates and infoboxes. In DBpedia, entities are mapped to a coarse-grained ontology which is collaboratively maintained and contains only about 270 classes corresponding to popular named entity types, in contrast to our goal of structuring the full set of Wikipedia articles in a larger and finer-grained taxonomy.

A few notable efforts to reconcile the two sides of Wikipedia, i.e., pages and categories, have been put forward very recently: WikiNet (Nastase et al., 2010; Nastase and Strube, 2013) is a project which heuristically exploits different aspects of Wikipedia to obtain a multilingual concept network by deriving not only is-a relations, but also other types of relations. A second project, MENTA (de Melo and Weikum, 2010), creates one of the largest multilingual lexical knowledge bases by interconnecting more than 13M articles in 271 languages. In contrast to our work, hypernym extraction is supervised in that decisions are made on the basis of labelled training examples and requires a reconciliation step owing to the heterogeneous nature of the hypernyms, something that we only do for categories, due to their noisy network. While WikiNet and MENTA bring together the knowledge available both at the page and category level, like we do, they either achieve low precision and coverage of the taxonomical structure or exhibit overly general hypernyms, as we show in our experiments in the next section.

Our work differs from the others in at least three respects: first, in marked contrast to most other resources, but similarly to WikiNet and WikiTaxonomy, our resource is self-contained and does not depend on other resources such as WordNet; second, we address the taxonomization task on both sides, i.e., pages and categories, by providing an algorithm which mutually and iteratively transfers knowledge from one side of the bitaxonomy to the other; third, we provide a wide coverage bitaxonomy closer in structure and granularity to a manual WordNet-like taxonomy, in contrast, for example, to DBpedia's flat entity-focused hierarchy.[2]

[2]Note that all the competitors on categories have average height between 1 and 3.69 on their biggest component, while we have 23.26, while on pages their height is between 1.9 and 4.22, while ours is 6.98. Since WordNet's average height is 8.07 we deem WiBi to be the resource structurally closest to WordNet.

Dataset	System	Prec.	Rec.	Cov.
Pages	WiBi	84.11	79.40	92.57
	WikiNet	57.29††	71.45††	82.01
	DBpedia	87.06	51.50††	55.93
	MENTA	81.52	72.49†	88.92
Categories	WiBi	85.18	82.88	97.31
	WikiTax	88.50	54.83††	59.43
	YAGO	94.13	53.41††	56.74
	MENTA	87.11	84.63	97.15
	MENTA$^{-\text{ENT}}$	85.18	71.95††	84.47

Table 2: Page and category taxonomy evaluation. † (††) denotes statistically significant difference, using χ^2 test, $p < 0.02$ ($p < 0.01$) between WiBi and the daggered resource.

7 Comparative Evaluation

7.1 Experimental Setup

We compared our resource (WiBi) against the Wikipedia taxonomies of the major knowledge resources in the literature providing hypernym links, namely DBpedia, WikiNet, MENTA, WikiTaxonomy and YAGO (see Section 6). As datasets, we used our gold standards of 1,000 randomly-sampled pages (see Section 3.3) and categories (see Section 5.1). In order to ensure a level playing field, we detected those pages (categories) which do not exist in any of the above resources and removed them to ensure full coverage of the dataset across all resources. For each resource we calculated precision, by manually marking each hypernym returned for each page (category) as correct or not. As regards recall, we note that in two cases (i.e., DBpedia returning page supertypes from its upper taxonomy, YAGO linking categories to WordNet synsets) the generalizations are neither pages nor categories and that MENTA returns heterogeneous hypernyms as mixed sets of WordNet synsets, Wikipedia pages and categories. Given this heterogeneity, standard recall across resources could not be calculated. For this reason we calculated recall as described in Section 3.3.

7.2 Results

Wikipedia pages We first report the results of the knowledge resources which provide page hypernyms, i.e., we compare against WikiNet, DBpedia and MENTA. We use the original outputs from the three resources: the first two are based on dumps which are from the same year as the one used in WiBi (cf. Section 3.3), while MENTA is based on a dump dating back to 2010 (consisting of 3.25M pages and 565k categories). We decided to include the latter for comparison purposes, as it uses knowledge from 271 Wikipedias to build the final taxonomy. However, we recognize its performance might be relatively higher on a 2012 dump.

We show the results on our page hypernym dataset in Table 2 (top). As can be seen, WikiNet obtains the lowest precision, due to the high number of hypernyms provided, many of which are incorrect, with a recall between that of DBpedia and MENTA. WiBi outperforms all other resources with 84.11% precision, 79.40% recall and 92.57% coverage. MENTA seems to be the closest resource to ours, however, we remark that the hypernyms output by MENTA are very heterogeneous: 48% of answers are represented by a WordNet synset, 37% by Wikipedia categories and 15% are Wikipedia pages. In contrast to all other resources, WiBi outputs page hypernyms only.

Wikipedia categories We then compared all the knowledge resources which deal with categories, i.e., WikiTaxonomy, YAGO and MENTA. For the latter two, the above considerations about the 2012 dump hold, whereas we reimplemented WikiTaxonomy, which was based on a 2009 dump, to run it on the same dump as WiBi. We excluded WikiNet from our comparison because it turned out to have low coverage of categories (i.e., less than 1%).

We show the results on our category dataset in Table 2 (bottom). Despite other systems exhibiting higher precision, WiBi generally achieves higher recall, thanks also to its higher category coverage. YAGO obtains the lowest recall and coverage, because only leaf categories are considered. MENTA is the closest system to ours, obtaining slightly higher precision and recall. Notably, however, MENTA outputs the first WordNet sense of *entity* for 13.17% of all the given answers, which, despite being correct and accounted in precision and recall, is uninformative. Since a system which always outputs *entity* would maximise all the three measures, we also calculated the performance for MENTA when discarding *entity* as an answer; as Table 2 shows (bottom, MENTA$^{-\text{ENT}}$), recall drops to 71.95%. Further analysis, presented below, shows that the specificity of its hypernyms is considerably lower than that of WiBi.

7.3 Analysis of the results

To get further insight into our results we performed two additional analyses of the data. First, we estimated the level of specialization of the hypernyms in the different resources on our two datasets. The idea is that a hypernym should be

Dataset	System (X)	WiBi=X	WiBi>X	WiBi<X
Pages	WikiNet	33.38	34.94	31.68
	DBpedia	31.68	56.71	11.60
	MENTA	19.04	50.85	30.12
Categories	WikiTax	43.11	38.51	18.38
	YAGO	12.36	81.14	6.50
	MENTA	12.36	73.69	13.95

Table 3: Specificity comparison.

valid while at the same time being as specific as possible (e.g., SINGER should be preferred over PERSON). We therefore calculated a measure, which we called specificity, that computes the percentage of times a system outputs a more specific answer than another system. To do this, we annotated each hypernym returned by a system as follows: -1 if the answer was wrong, 0 if missing, $>$ 0 if correct; more specific answers were assigned higher scores. When comparing two systems, we select the respective most specific answers a_1, a_2 and say the first system is more specific than the latter whenever $score(a_1) > score(a_2)$. Table 3 shows the results for all the resources and for both the page and category taxonomies: WiBi consistently provides considerably more specific hypernyms than any other resource (middle column).

A second important aspect that we analyzed was the granularity of each taxonomy, determined by drawing each resource on a bidimensional plane with the number of distinct hypernyms on the x axis and the total number of hypernyms (i.e., edges) in the taxonomy on the y axis. Figures 5a and 5b show the position of each resource for the page and the category taxonomies, respectively. As can be seen, WiBi, as well as the page taxonomy of MENTA, is the resource with the best granularity, as not only does it attain high coverage, but it also provides a larger variety of classes as generalizations of pages and categories. Specifically, WiBi provides over 3M hypernym pages chosen from a range of 94k distinct hypernyms, while others exhibit a considerably smaller range of distinct hypernyms (e.g., DBpedia by design, but also WikiNet, with around 11k distinct page hypernyms). The large variety of classes provided by MENTA, however, is due to including more than 100k Wikipedia categories (among which, categories about *deaths* and *births* alone represent about 2% of the distinct hypernyms). As regards categories, while the number of distinct hypernyms of WiBi and WikiTaxonomy is approximately the same (around 130k), the total number of hypernyms (around 580k for both taxonomies) is distributed over half of the categories in WikiTaxonomy compared to WiBi, resulting in a double number of hypernyms per category, but lower coverage (cf. Table 2).

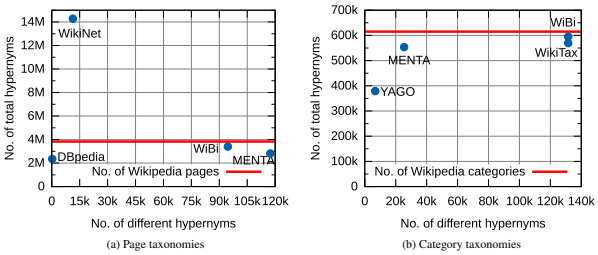


Figure 5: Hypernym granularity for the resources.

8 Conclusions

In this paper we have presented WiBi, an automatic 3-phase approach to the construction of a bitaxonomy for the English Wikipedia, i.e., a full-fledged, integrated page and category taxonomy: first, using a set of high-precision linkers, the page taxonomy is populated; next, a fixed point algorithm populates the category taxonomy while enriching the page taxonomy iteratively; finally, the category taxonomy undergoes structural refinements. Coverage, quality and granularity of the bitaxonomy are considerably higher than the taxonomy of state-of-the-art resources like DBpedia, YAGO, MENTA, WikiNet and WikiTaxonomy.

Our contributions are three-fold: i) we propose a unified, effective approach to the construction of a Wikipedia bitaxonomy, a richer structure than those produced in the literature; ii) our method for building the bitaxonomy is self-contained, thanks to its independence from external resources (like WordNet) and the virtual absence of supervision, making WiBi replicable on any new version of Wikipedia; iii) the taxonomy provides nearly full coverage of pages and categories, encompassing the entire encyclopedic knowledge in Wikipedia.

We will apply our video games with a purpose (Vannella et al., 2014) to validate WiBi. We also plan to integrate WiBi into BabelNet (Navigli and Ponzetto, 2012), so as to fully taxonomize it, and exploit its high quality for improving semantic predicates (Flati and Navigli, 2013).

Acknowledgments

The authors gratefully acknowledge the support of the ERC Starting Grant MultiJEDI No. 259234.
We thank Luca Telesca for his implementation of WikiTaxonomy and Jim McManus for his comments on the manuscript.

References

Robert A. Amsler. 1981. A Taxonomy for English Nouns and Verbs. In *Proceedings of Association for Computational Linguistics (ACL '81)*, pages 133–138, Stanford, California, USA.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ive. 2007. DBpedia: A nucleus for a web of open data. In *Proceedings of 6th International Semantic Web Conference joint with 2nd Asian Semantic Web Conference (ISWC+ASWC 2007)*, pages 722–735, Busan, Korea.

Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. DBpedia - a crystallization point for the Web of Data. *Web Semantics*, 7(3):154–165.

Sebastian Blohm. 2007. Using the web to reduce data sparseness in pattern-based information extraction. In *Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 18–29, Warsaw, Poland. Springer.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the International Conference on Management of Data (SIGMOD '08)*, SIGMOD '08, pages 1247–1250, New York, NY, USA.

Nicoletta Calzolari, Laura Pecchia, and Antonio Zampolli. 1973. Working on the Italian Machine Dictionary: a Semantic Approach. In *Proceedings of the 5th Conference on Computational Linguistics (COLING '73)*, pages 49–70, Pisa, Italy.

Nicoletta Calzolari. 1982. Towards the organization of lexical definitions on a database structure. In *Proc. of the 9th Conference on Computational Linguistics (COLING '82)*, pages 61–64, Prague, Czechoslovakia.

Gerard de Melo and Gerhard Weikum. 2010. MENTA: Inducing Multilingual Taxonomies from Wikipedia. In *Proceedings of Conference on Information and Knowledge Management (CIKM '10)*, pages 1099–1108, New York, NY, USA.

Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2004. Web-scale information extraction in knowItAll: (preliminary results). In *Proceedings of the 13th International Conference on World Wide Web (WWW '04)*, pages 100–110, New York, NY, USA. ACM.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Database*. MIT Press, Cambridge, MA.

David A. Ferrucci. 2012. Introduction to "This is Watson". *IBM Journal of Research and Development*, 56(3):1.

Tiziano Flati and Roberto Navigli. 2013. SPred: Large-scale Harvesting of Semantic Predicates. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1222–1232, Sofia, Bulgaria.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the International Conference on Computational Linguistics (COLING '92)*, pages 539–545, Nantes, France.

Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence*, 194:28–61.

Eduard H. Hovy, Roberto Navigli, and Simone Paolo Ponzetto. 2013. Collaboratively built semi-structured content and Artificial Intelligence: The story so far. *Artificial Intelligence*, 194:2–27.

Nancy Ide and Jean Véronis. 1993. Extracting knowledge bases from machine-readable dictionaries: Have we wasted our time? In *Proceedings of the Workshop on Knowledge Bases and Knowledge Structures*, pages 257–266, Tokyo, Japan.

Dan Klein and Christopher D. Manning. 2003. Fast Exact Inference with a Factored Model for Natural Language Parsing. In *Advances in Neural Information Processing Systems 15 (NIPS)*, pages 3–10, Vancouver, British Columbia, Canada.

Zornitsa Kozareva and Eduard H. Hovy. 2010. A Semi-Supervised Method to Learn and Construct Taxonomies Using the Web. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '10)*, pages 1110–1118, Seattle, WA, USA.

Olena Medelyan, David Milne, Catherine Legg, and Ian H. Witten. 2009. Mining meaning from Wikipedia. *International Journal of Human-Computer Studies*, 67(9):716–754.

Shachar Mirkin, Ido Dagan, and Eyal Shnarch. 2009. Evaluating the inferential utility of lexical-semantic resources. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 558–566, Athens, Greece.

Tom Mitchell. 2005. Reading the Web: A Breakthrough Goal for AI. *AI Magazine*.

Vivi Nastase and Michael Strube. 2013. Transforming Wikipedia into a large scale multilingual concept network. *Artificial Intelligence*, 194:62–85.

Vivi Nastase, Michael Strube, Benjamin Boerschinger, Caecilia Zirn, and Anas Elghafari. 2010. WikiNet: A Very Large Scale Multi-Lingual Concept Network. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta.

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

Roberto Navigli and Paola Velardi. 2010. Learning Word-Class Lattices for Definition and Hypernym Extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)*, pages 1318–1327, Uppsala, Sweden, July. Association for Computational Linguistics.

Patrick Pantel and Deepak Ravichandran. 2004. Automatically labeling semantic classes. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT 2013), Boston, Massachusetts, 2–7 May 2004*, pages 321–328.

Simone Paolo Ponzetto and Michael Strube. 2007. Deriving a large scale taxonomy from Wikipedia. In *Proceedings of the 22nd Conference on the Advancement of Artificial Intelligence (AAAI '07), Vancouver, B.C., Canada, 22–26 July 2007*, pages 1440–1445.

Simone Paolo Ponzetto and Michael Strube. 2011. Taxonomy induction based on a collaboratively built knowledge repository. *Artificial Intelligence*, 175(9-10):1737–1756.

Hoifung Poon, Janara Christensen, Pedro Domingos, Oren Etzioni, Raphael Hoffmann, Chloe Kiddon, Thomas Lin, Xiao Ling, Mausam, Alan Ritter, Stefan Schoenmackers, Stephen Soderland, Dan Weld, Fei Wu, and Congle Zhang. 2010. Machine Reading at the University of Washington. In *Proceedings of the 1st International Workshop on Formalisms and Methodology for Learning by Reading in conjunction with NAACL-HLT 2010*, pages 87–95, Los Angeles, California, USA.

Maria Ruiz-Casado, Enrique Alfonseca, and Pablo Castells. 2005. Automatic assignment of Wikipedia encyclopedic entries to WordNet synsets. In *Advances in Web Intelligence*, volume 3528 of *Lecture Notes in Computer Science*, pages 380–386. Springer Verlag.

Amit Singhal. 2012. Introducing the Knowledge Graph: Things, Not Strings. Technical report, Official Blog (of Google). Retrieved May 18, 2012.

Rion Snow, Dan Jurafsky, and Andrew Ng. 2006. Semantic taxonomy induction from heterogeneous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL 2006)*, pages 801–808.

Fabian Suchanek and Gerhard Weikum. 2013. Knowledge harvesting from text and Web sources. In *IEEE 29th International Conference on Data Engineering (ICDE 2013)*, pages 1250–1253, Brisbane, Australia. IEEE Computer Society.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2008. YAGO: A large ontology from Wikipedia and WordNet. *Journal of Web Semantics*, 6(3):203–217.

Daniele Vannella, David Jurgens, Daniele Scarfini, Domenico Toscani, and Roberto Navigli. 2014. Validating and Extending Semantic Knowledge Bases using Video Games with a Purpose. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)*, Baltimore, USA.

Paola Velardi, Stefano Faralli, and Roberto Navigli. 2013. OntoLearn Reloaded: A graph-based algorithm for taxonomy induction. *Computational Linguistics*, 39(3):665–707.

Information Extraction over Structured Data: Question Answering with Freebase

Xuchen Yao [1] and **Benjamin Van Durme** [1,2]
[1]Center for Language and Speech Processing
[2]Human Language Technology Center of Excellence
Johns Hopkins University
Baltimore, MD, USA

Abstract

Answering natural language questions using the Freebase knowledge base has recently been explored as a platform for advancing the state of the art in open domain semantic parsing. Those efforts map questions to sophisticated meaning representations that are then attempted to be matched against viable answer candidates in the knowledge base. Here we show that relatively modest information extraction techniques, when paired with a web-scale corpus, can outperform these sophisticated approaches by roughly 34% relative gain.

1 Introduction

Question answering (QA) from a knowledge base (KB) has a long history within natural language processing, going back to the 1960s and 1970s, with systems such as `Baseball` (Green Jr et al., 1961) and `Lunar` (Woods, 1977). These systems were limited to closed-domains due to a lack of knowledge resources, computing power, and ability to robustly understand natural language. With the recent growth in KBs such as `DBPedia` (Auer et al., 2007), `Freebase` (Bollacker et al., 2008) and `Yago2` (Hoffart et al., 2011), it has become more practical to consider answering questions across wider domains, with commercial systems including `Google Now`, based on Google's `Knowledge Graph`, and `Facebook Graph Search`, based on social network connections.

The AI community has tended to approach this problem with a focus on first understanding the intent of the question, via shallow or deep forms of semantic parsing (c.f. §3 for a discussion). Typically questions are converted into some meaning representation (e.g., the lambda calculus), then mapped to database queries. Performance is thus bounded by the accuracy of the original semantic parsing, and the well-formedness of resultant database queries.[1]

The Information Extraction (IE) community approaches QA differently: first performing relatively coarse information retrieval as a way to triage the set of possible answer candidates, and only then attempting to perform deeper analysis.

Researchers in semantic parsing have recently explored QA over Freebase as a way of moving beyond closed domains such as GeoQuery (Tang and Mooney, 2001). While making semantic parsing more robust is a laudable goal, here we provide a more rigorous IE baseline against which those efforts should be compared: we show that "traditional" IE methodology can significantly outperform prior state-of-the-art as reported in the semantic parsing literature, with a relative gain of 34% F_1 as compared to Berant et al. (2013).

2 Approach

We will view a KB as an interlinked collection of "topics". When given a question about one or several topics, we can select a "view" of the KB concerning only involved topics, then inspect every related node within a few hops of relations to the topic node in order to extract the answer. We call such a view a *topic graph* and assume answers can be found within the graph. We aim to maximally automate the answer extraction process, by massively combining discriminative features for both the question and the topic graph. With a high performance learner we have found that a system with millions of features can be trained within hours, leading to intuitive, human interpretable features. For example, we learn that given a question concerning money, such as: what money is used in

[1]As an example, 50% of errors of the CCG-backed (Kwiatkowski et al., 2013) system were contributed by parsing or structural matching failure.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 956–966,
Baltimore, Maryland, USA, June 23-25 2014.

ukraine, the expected answer type is likely currency. We formalize this approach in §4.

One challenge for natural language querying against a KB is the relative informality of queries as compared to the grammar of a KB. For example, for the question: who cheated on celebrity A, answers can be retrieved via the Freebase relation celebrity.infidelity.participant, but the connection between the phrase cheated on and the formal KB relation is not explicit. To alleviate this problem, the best attempt so far is to map from ReVerb (Fader et al., 2011) predicate-argument triples to Freebase relation triples (Cai and Yates, 2013; Berant et al., 2013). Note that to boost precision, ReVerb has already pruned down less frequent or credible triples, yielding not as much coverage as its text source, ClueWeb. Here we instead directly mine relation mappings from ClueWeb and show that both direct relation mapping precision and indirect QA F_1 improve by a large margin. Details in §5.

Finally, we tested our system, jacana-freebase,[2] on a realistic dataset generously contributed by Berant et al. (2013), who collected thousands of commonly asked questions by crawling the Google Suggest service. Our method achieves state-of-the-art performance with F_1 at 42.0%, a 34% relative increase from the previous F_1 of 31.4%.

3 Background

QA from a KB faces two prominent challenges: model and data. The model challenge involves finding the best meaning representation for the question, converting it into a query and executing the query on the KB. Most work approaches this via the bridge of various intermediate representations, including combinatory categorial grammar (Zettlemoyer and Collins, 2005, 2007, 2009; Kwiatkowski et al., 2010, 2011, 2013), synchronous context-free grammars (Wong and Mooney, 2007), dependency trees (Liang et al., 2011; Berant et al., 2013), string kernels (Kate and Mooney, 2006; Chen and Mooney, 2011), and tree transducers (Jones et al., 2012). These works successfully showed their effectiveness in QA, despite the fact that most of them require hand-labeled logic annotations. More recent research started to minimize this direct supervision by using latent meaning representations (Berant et al., 2013; Kwiatkowski et al., 2013) or distant supervision (Krishnamurthy and Mitchell, 2012).

We instead attack the problem of QA from a KB from an IE perspective: we learn directly the pattern of QA pairs, represented by the dependency parse of questions and the Freebase structure of answer candidates, without the use of intermediate, general purpose meaning representations.

The data challenge is more formally framed as ontology or (textual) schema matching (Hobbs, 1985; Rahm and Bernstein, 2001; Euzenat and Shvaiko, 2007): matching structure of two ontologies/databases or (in extension) mapping between KB relations and NL text. In terms of the latter, Cai and Yates (2013) and Berant et al. (2013) applied pattern matching and relation intersection between Freebase relations and predicate-argument triples from the ReVerb OpenIE system (Fader et al., 2011). Kwiatkowski et al. (2013) expanded their CCG lexicon with Wiktionary word tags towards more domain independence. Fader et al. (2013) learned question paraphrases from aligning multiple questions with the same answers generated by WikiAnswers. The key factor to their success is to have a huge text source. Our work pushes the data challenge to the limit by mining directly from ClueWeb, a 5TB collection of web data.

Finally, the KB community has developed other means for QA without semantic parsing (Lopez et al., 2005; Frank et al., 2007; Unger et al., 2012; Yahya et al., 2012; Shekarpour et al., 2013). Most of these work executed SPARQL queries on interlinked data represented by RDF (Resource Description Framework) triples, or simply performed triple matching. Heuristics and manual templates were also commonly used (Chu-Carroll et al., 2012). We propose instead to learn discriminative features from the data with shallow question analysis. The final system captures intuitive patterns of QA pairs automatically.

4 Graph Features

Our model is inspired by an intuition on how everyday people search for answers. If you asked someone: what is the name of justin bieber brother,[3] and gave them access to Freebase, that person might first determine that the question

[2] https://code.google.com/p/jacana

[3] All examples used in this paper come from the training data crawled from Google Suggest. They are lowercased and some contain typos.

is about Justin Bieber (or his brother), go to Justin Bieber's Freebase page, and search for his brother's name. Unfortunately Freebase does not contain an exact relation called brother, but instead sibling. Thus further inference (i.e., brother $\leftrightarrow$ male sibling) has to be made. In the following we describe how we represent this process.

4.1 Question Graph

In answering our example query a person might take into consideration multiple constraints. With regards to the question, we know we are looking for the name of a person based on the following:

- the dependency relation nsubj(what, name) and prep_of(name, brother) indicates that the question seeks the information of a name;[4]
- the dependency relation prep_of(name, brother) indicates that the name is about a brother (but we do not know whether it is a person name yet);
- the dependency relation nn(brother, bieber) and the facts that, (i) Bieber is a person and (ii) a person's brother should also be a person, indicate that the name is about a person.

This motivates the design of dependency-based features. We show one example in Figure 1(a), left side. The following linguistic information is of interest:

- question word (*qword*), such as what/who/how many. We use a list of 9 common qwords. [5]
- question focus (*qfocus*), a cue of expected answer types, such as name/money/time. We keep our analysis simple and do not use a question classifier, but simply extract the noun dependent of qword as qfocus.
- question verb (*qverb*), such as is/play/take, extracted from the main verb of the question. Question verbs are also good hints of answer types. For instance, play is likely to be followed by an instrument, a movie or a sports team.
- question topic (*qtopic*). The topic of the question helps us find relevant Freebase pages. We simply apply a named entity recognizer to find the question topic. Note that there can be more than one topic in the question.

Then we convert the dependency parse into a more generic question graph, in the following steps:

1. if a node was tagged with a question feature, then replace this node with its question feature, e.g., what $\rightarrow$ qword=what;
2. (special case) if a qtopic node was tagged as a named entity, then replace this node with its its named entity form, e.g., bieber $\rightarrow$ qtopic=person;
3. drop any leaf node that is a determiner, preposition or punctuation.

The converted graph is shown in Figure 1(a), right side. We call this a *question feature graph*, with every node and relation a potential feature for this question. Then features are extracted in the following form: with s the source and t the target node, for every edge $e(s,t)$ in the graph, extract s, t, $s \mid t$ and $s \mid e \mid t$ as features. For the edge, prep_of(qfocus=name, brother), this would mean the following features: qfocus=name, brother, qfocus=name|brother, and qfocus=name|prep_of|brother.

We show with examples why these features make sense later in §6 Table 6. Furthermore, the reason that we have kept some lexical features, such as brother, is that we hope to learn from training a high correlation between brother and some Freebase relations and properties (such as sibling and male) if we do not possess an external resource to help us identify such a correlation.

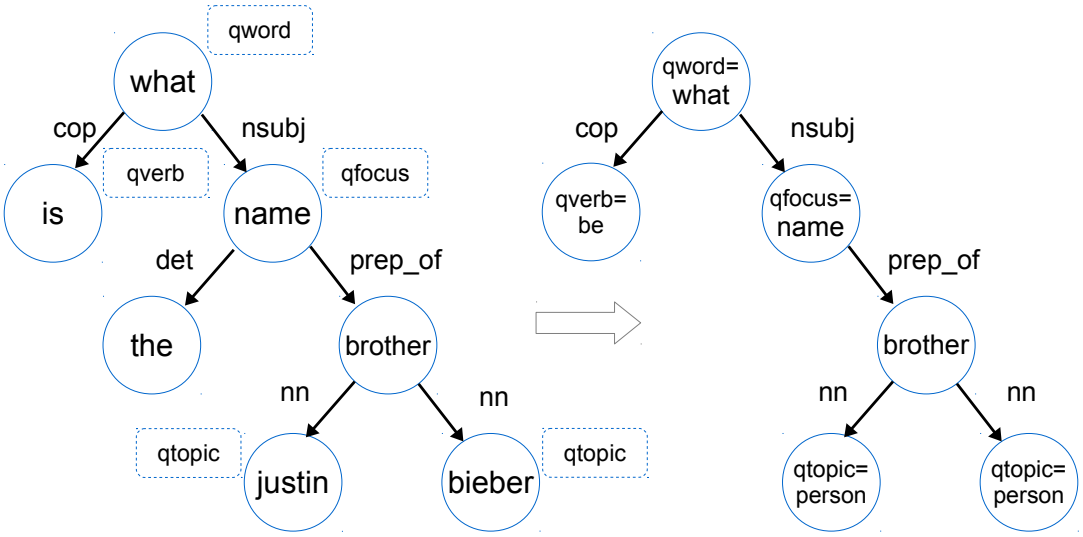
4.2 Freebase Topic Graph

Given a topic, we selectively roll out the Freebase graph by choosing those nodes within a few hops of relationship to the *topic node*, and form a *topic graph*. Besides incoming and/or outgoing relationships, nodes also have *properties*: a string that describes the attribute of a node, for instance, node type, gender or height (for a person). One major difference between relations and properties is that both arguments of a relation are nodes, while only one argument of a property is a node, the other a string. Arguments of relations are usually interconnected, e.g., London can be the place_of_birth for Justin Bieber, or capital_of the UK. Arguments of properties are attributes that are only "attached" to certain nodes and have no outgoing edges. Figure 1(b) shows an example.

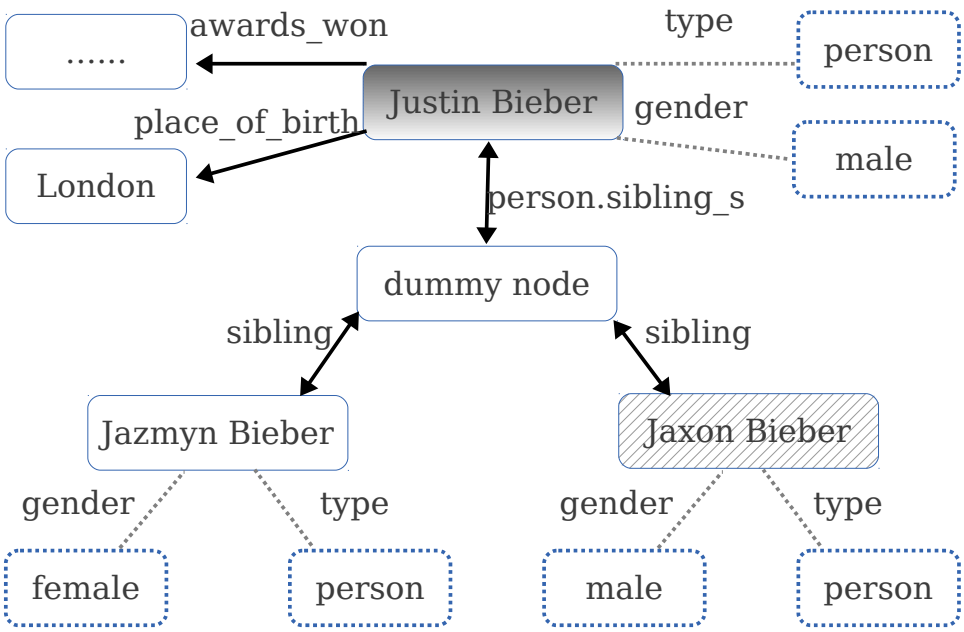
Both relationship and property of a node are important to identifying the answer. They connect the nodes with the question and describe some unique characteristics. For instance, without the properties type:person and gender:male,

[4] We use the Stanford collapsed dependency form.

[5] who, when, what, where, how, which, why, whom, whose.



(a) Dependence parse with annotated question features in dashed boxes (left) and converted feature graph (right) with only relevant and general information about the original question kept. Note that the left is a real but incorrect parse.



(b) A view of Freebase graph on the Justin Bieber topic with nodes in solid boxes and properties in dashed boxes. The hatching node, Jaxon Bieber, is the answer. Freebase uses a dummy parent node for a list of nodes with the same relation.

Figure 1: Dependency parse and excerpted Freebase topic graph on the question what is the name of justin bieber brother.

we would not have known the node Jaxon Bieber represents a male person. These properties, along with the sibling relationship to the topic node, are important cues for answering the question. Thus for the Freebase graph, we use relations (with directions) and properties as features for each node.

Additionally, we have analyzed how Freebase relations map back to the question. Some of the mapping can be simply detected as paraphrasing or lexical overlap. For example, the person.parents relationship helps answering questions about parenthood. However, most Freebase relations are framed in a way that is not commonly addressed in natural language questions. For instance, for common celebrity gossip questions like who cheated on celebrity A, it is hard for a system to find the Freebase relation celebrity.infidelity.participant as the target relation if it had not observed this pattern in training.

Thus assuming there is an alignment model that is able to tell how likely one relation maps to the original question, we add extra alignment-based features for the incoming and outgoing relation of each node. Specifically, for each relation rel in a topic graph, we compute $P(rel \mid question)$ to rank the relations. Finally the ranking (e.g., top 1/2/5/10/100 and beyond) of each relation is used as features instead of a pure probability. We describe such an alignment model in § 5.

4.3 Feature Production

We combine question features and Freebase features (per node) by doing a pairwise concatenation. In this way we hope to capture the association between question patterns and answer nodes. For instance, in a loglinear model setting, we expect to learn a high feature weight for features like:

qfocus=money|node_type=currency

and a very low weight for:

qfocus=money|node_type=person.

This combination greatly enlarges the total number of features, but owing to progress in large-scale machine learning such feature spaces are less of a concern than they once were (concrete numbers in § 6 Model Tuning).

5 Relation Mapping

In this section we describe a "translation" table between Freebase relations and NL words was built.

5.1 Formula

The objective is to find the most likely relation a question prompts. For instance, for the question who is the father of King George VI, the most likely relation we look for is people.person.parents. To put it more formally, given a question Q of a word vector $\mathbf{w}$, we want to find out the relation R that maximizes the probability $P(R \mid Q)$.

More interestingly, for the question who is the father of the Periodic Table, the actual relation that encodes its original meaning is law.invention.inventor, rather than people.person.parents. This simple example points out that *every* part of the question could change what the question inquires eventually. Thus we need to count for *each* word w in Q. Due to the bias and incompleteness of any data source, we approximate the true probability of P with $\tilde{P}$ under our specific model. For the simplicity of computation, we assume conditional independence between words and apply Naive Bayes:

$$\begin{aligned}\tilde{P}(R \mid Q) &\propto \tilde{P}(Q \mid R)\tilde{P}(R) \\ &\approx \tilde{P}(\mathbf{w} \mid R)\tilde{P}(R) \\ &\approx \prod_w \tilde{P}(w \mid R)\tilde{P}(R)\end{aligned}$$

where $\tilde{P}(R)$ is the prior probability of a relation R and $\tilde{P}(w \mid R)$ is the conditional probability of word w given R.

It is possible that we do not observe a certain relation R when computing the above equation. In this case we back off to the "sub-relations": a relation R is a concatenation of a series of sub-relations $R = \mathbf{r} = r_1.r_2.r_3.\ldots$. For instance, the sub-relations of people.person.parents are people, person, and parents. Again, we assume conditional independence between sub-relations and apply Naive Bayes:

$$\begin{aligned}\tilde{P}_{\text{backoff}}(R \mid Q) &\approx \tilde{P}(\mathbf{r} \mid Q) \\ &\approx \prod_r \tilde{P}(r \mid Q) \\ &\propto \prod_r \tilde{P}(Q \mid r)\tilde{P}(r) \\ &\approx \prod_r \prod_w \tilde{P}(w \mid r)\tilde{P}(r)\end{aligned}$$

One other reason that we estimated $\tilde{P}(w \mid r)$ and $\tilde{P}(r)$ for sub-relations is that Freebase relations share some common structures in between them. For instance, both people.person.parents and fictional_universe.fictional_character.parents

indicate the parent relationship but the latter is much less commonly annotated. We hope that the shared sub-relation, parents, can help better estimate for the less annotated. Note that the backoff model would have a much smaller value than the original, due to double multiplication $\prod_r \prod_w$. In practice we normalize it by the sub-relations size to keep it at the same scale with $\tilde{P}(R \mid Q)$.

Finally, to estimate the prior and conditional probability, we need a massive data collection.

5.2 Steps

The ClueWeb09[6] dataset is a collection of 1 billion webpages (5TB compressed in raw HTML) in 10 languages by Carnegie Mellon University in 2009. FACC1, the Freebase Annotation of the ClueWeb Corpus version 1 (Gabrilovich et al., 2013), contains index and offset of Freebase entities within the English portion of ClueWeb. Out of all 500 million English documents, 340 million were automatically annotated with at least one entity, with an average of 15 entity mentions per document. The precision and recall of annotation were estimated at $80-85\%$ and $70-85\%$ (Orr et al., 2013).

Given these two resources, for each binary Freebase relation, we can find a collection of sentences each of which contains both of its arguments, then simply learn how words in these sentences are associated with this relation, i.e., $\tilde{P}(w \mid R)$ and $\tilde{P}(w \mid r)$. By counting how many times each relation R was annotated, we can estimate $\tilde{P}(R)$ and $\tilde{P}(r)$. The learning task can be framed in the following short steps:

1. We split each HTML document by sentences (Kiss and Strunk, 2006) using NLTK (Bird and Loper, 2004) and extracted those with at least two Freebase entities which has at least one direct established relation according to Freebase.
2. The extraction formed two parallel corpora, one with "relation - sentence" pairs (for estimating $\tilde{P}(w \mid R)$ and $\tilde{P}(R)$) and the other with "subrelations - sentence" pairs (for $\tilde{P}(w \mid r)$ and $\tilde{P}(r)$). Each corpus has 1.2 billion pairs.
3. The tricky part was to align these 1.2 billion pairs. Since the relations on one side of these pairs are not *natural* sentences, we ran the most simple IBM alignment Model 1 (Brown et al., 1993) to estimate the translation probability with GIZA++ (Och and Ney, 2003). To speed up, the 1.2 billion pairs were split into 100 even chunks. We ran 5 iterations of EM on each one and finally aligned the 1.2 billion pairs from both directions. To symmetrize the alignment, common MT heuristics INTERSECTION, UNION, GROW-DIAG-FINAL, and GROW-DIAG-FINAL-AND (Koehn, 2010) were separately applied and evaluated later.
4. Treating the aligned pairs as *observation*, the co-occurrence matrix between aligning relations and words was computed. There were 10,484 relations and sub-relations in all, and we kept the top 20,000 words.
5. From the co-occurrence matrix we computed $\tilde{P}(w \mid R)$, $\tilde{P}(R)$, $\tilde{P}(w \mid r)$ and $\tilde{P}(r)$.

[6] http://lemurproject.org/clueweb09/

$\mathbf{0}$	$\leq \mathbf{10}$	$\leq \mathbf{10^2}$	$\leq \mathbf{10^3}$	$\leq \mathbf{10^4}$	$> \mathbf{10^4}$
7.0%	0.7%	1.2%	0.4%	1.3%	89.5%

Table 1: Percentage of answer relations (the incoming relation connected to the answer node) with respect to how many sentences we learned this relation from in CluewebMapping. For instance, the first column says there are 7% of answer relations for which we cannot find a mapping (so we had to use the backoff probability estimation); the last column says there are 89.5% of answer relations that we were able to learn the mapping between this relation and text based on more than 10 thousand relation-sentence pairs. The total number of answer relations is 7886.

Hand-checking the learned probabilities shows both success, failure and some bias. For instance, for the film.actor.film relation (mapping from film names to actor names), the top words given by $\tilde{P}(w \mid R)$ are won, star, among, show. For the film.film.directed_by relation, some important stop words that could indicate this relation, such as by and with, rank directly after director and direct. However, due to significant popular interest in certain news categories, and the resultant catering by websites to those information desires, then for example we also learned a heavily correlated connection between Jennifer Aniston and celebrity.infidelity.victim, and between some other you-know-who names and celebrity.infidelity.participant.

We next formally evaluate how the learned mapping help predict relations from words.

5.3 Evaluation

Both ClueWeb and its Freebase annotation has a bias. Thus we were firstly interested in the coverage of mined relation mappings. As a comparison, we used a dataset of relation mapping contributed by Berant et al. (2013) and Lin et al. (2012). The idea is very similar: they intersected Freebase relations with predicates in (arg1, predicate, arg2) triples extracted from `ReVerb` to learn the mapping between Freebase relations and triple predicates. Note the scale difference: although `ReVerb` was also extracted from ClueWeb09, there were only 15 million triples to intersect with the relations, while we had 1.2 billion alignment pairs. We call this dataset **ReverbMapping** and ours **CluewebMapping**.

The evaluation dataset, WEBQUESTIONS, was also contributed by Berant et al. (2013). It contains 3778 training and 2032 test questions collected from the Google Suggest service. All questions were annotated with answers from Freebase. Some questions have more than one answer, such as what to see near sedona arizona?.

We evaluated on the training set in two aspects: coverage and prediction performance. We define *answer node* as the node that is the answer and *answer relation* as the relation from the answer node to its direct parent. Then we computed how much and how well the answer relation was triggered by ReverbMapping and CluewebMapping. Thus for the question, who is the father of King George VI, we ask two questions: does the mapping, 1. (coverage) contain the answer relation people.person.parents? 2. (precision) predict the answer relation from the question?

Table 1 shows the coverage of CluewebMapping, which covers 93.0% of all answer relations. Among them, we were able to learn the rule mapping using more than 10 thousand relation-sentence pairs for *each* of the 89.5% of all answer relations. In contrast, ReverbMapping covers 89.7% of the answer relations.

Next we evaluated the prediction performance, using the evaluation metrics of information retrieval. For each question, we extracted all relations in its corresponding topic graph, and ranked each relation with whether it is the answer relation. For instance, for the previous example question, we want to rank the relation people.person.parents as number 1. We computed standard MAP (Mean Average Precision) and MRR (Mean Reciprocal Rank), shown in Table 2(a). As a simple baseline, "word overlap" counts the overlap between relations and the question. CluewebMapping ranks each relation by $\tilde{P}(R \mid Q)$. ReverbMapping does the same, except that we took a uniform distribution on $\tilde{P}(w \mid R)$ and $\tilde{P}(R)$ since the contributed dataset did not include co-occurrence counts to estimate these probabilities.[7] Note that the median rank from CluewebMapping is only 12, indicating that half of all answer relations are ranked in the top 12.

Table 2(b) further shows the percentage of answer relations with respect to their ranking. CluewebMapping successfully ranked 19% of answer relations as top 1. A sample of these includes person.place_of_birth, location.containedby, country.currency_used, regular_tv_appearance.actor, etc. These percentage numbers are good clue for feature design: for instance, we may be confident in a relation if it is ranked top 5 or 10 by CluewebMapping.

To conclude, we found that CluewebMapping provides satisfying coverage on the 3778 training questions: only 7% were missing, despite the biased nature of web data. Also, CluewebMapping gives reasonably good precision on its prediction, despite the noisy nature of web data. We move on to fully evaluate the final QA F_1.

6 Experiments

We evaluate the final F_1 in this section. The system of comparison is that of Berant et al. (2013).

Data We re-used WEBQUESTIONS, a dataset collected by Berant et al. (2013). It contains 5810 questions crawled from the Google Suggest service, with answers annotated on Amazon Mechanical Turk. All questions contain at least one answer from Freebase. This dataset has been split by $65\%/35\%$ into TRAIN-ALL and TEST. We further randomly divided TRAIN-ALL by $80\%/20\%$ to a smaller TRAIN and development set DEV. Note that our DEV set is different from that of Berant et al. (2013), but the final result on TEST is directly comparable. Results are reported in terms of macro F_1 with partial credit (following Berant et al. (2013)) if a predicted answer list does not have a perfect match with all gold answers, as a

[7] The way we used ReverbMapping was not how Berant et al. (2013) originally used it: they employed a discriminative log-linear model to judge relations and that might yield better performance. As a fair comparison, ranking of CluewebMapping under uniform distribution is also included in Table 2(a).

	Median Rank	MAP	MRR
word overlap	471	0.0380	0.0590
ReverbMapping	60	0.0691	0.0829
CluewebMapping	12	0.2074	0.2900
with uniform dist.	61	0.0544	0.0561

(a) Ranking on answer relations. Best result on CluewebMapping was under the GROW-DIAG-FINAL-AND heuristics (row 3) when symmetrizing alignment from both directions. The last row shows ranking of CluewebMapping under uniform distribution (assuming counting on words and relations is not known).

	1	≤ 5	≤ 10	≤ 50	≤ 100	> 100
w. o.	3.5	4.7	2.5	3.9	4.1	81.3
R.M.	2.6	9.1	8.6	26.0	13.0	40.7
C.M.	19.0	19.9	8.9	22.3	7.5	22.4

(b) Percentage of answer relations w.r.t. ranking number (header). w.o.: word overlap; R.M.: ReverbMapping; C.M.: CluewebMapping.

Table 2: Evaluation on answer relation ranking prediction on 3778 training questions.

lot of questions in WEBQUESTIONS contain more than one answer.

Search With an Information Retrieval (IR) front-end, we need to locate the exact Freebase topic node a question is about. For this purpose we used the Freebase Search API (Freebase, 2013a).All named entities [8] in a question were sent to this API, which returned a ranked list of relevant topics. We also evaluated how well the search API served the IR purpose. WEBQUESTIONS not only has answers annotated, but also which Freebase topic nodes the answers come from. Thus we evaluated the ranking of retrieval with the gold standard annotation on TRAIN-ALL, shown in Table 3. The top 2 results of the Search API contain gold standard topics for more than 90% of the questions and the top 10 results contain more than 95%. We took this as a "good enough" IR front-end and used it on TEST.

Once a topic is obtained we query the Freebase Topic API (Freebase, 2013b) to retrieve all relevant information, resulting in a topic graph. The API returns almost identical information as displayed via a web browser to a user viewing this topic. Given that turkers annotated answers based on the topic page via a browser, this supports the assumption that the same answer would be located in the topic graph, which is then passed to the QA engine for feature extraction and classification.

[8] When no named entities are detected, we fall back to noun phrases.

top	1	2	3	5	10
#	3263	3456	3532	3574	3604
%	86.4	91.5	93.5	94.6	95.4

Table 3: Evaluation on the Freebase Search API: how many questions' top n retrieved results contain the gold standard topic. Total number of questions is 3778 (size of TRAIN-ALL). There were only 5 questions with no retrieved results.

	P	R	F_1
basic	57.3	30.1	39.5
+ word overlap	56.0	31.4	40.2
+ CluewebMapping	59.9	35.4	44.5
+both	59.0	35.4	44.3

Table 4: F_1 on DEV with different feature settings.

Model Tuning We treat QA on Freebase as a binary classification task: for each node in the topic graph, we extract features and judge whether it is the answer node. Every question was processed by the Stanford CoreNLP suite with the caseless model. Then the question features (§4.1) and node features (§4.2) were combined (§4.3) for each node. The learning problem is challenging: for about 3000 questions in TRAIN, there are 3 million nodes (1000 nodes per topic graph), and 7 million feature types. We employed a high-performance machine learning tool, `Classias` (Okazaki, 2009). Training usually took around 4 hours. We experimented with various discriminative learners on DEV, including logistic regression, perceptron and SVM, and found L1 regularized logistic regression to give the best result. The L1 regularization encourages sparse features by driving feature weights towards zero, which was ideal for the over-generated feature space. After training, we had around 30 thousand features with non-zero weights, a 200 fold reduction from the original features.

Also, we did an ablation test on DEV about how additional features on the mapping between Freebase relations and the original questions help, with three feature settings: 1) "basic" features include feature productions read off from the feature graph (Figure 1); 2) "+ word overlap" adds additional features on whether sub-relations have overlap with the question; and 3) "+ CluewebMapping" adds the ranking of relation prediction given the question according to CluewebMapping. Table 4 shows that the additional CluewebMapping

	P	R	F_1
Gold Retrieval	45.4	52.2	48.6
Freebase Search API	38.8	45.8	42.0
Berant et al. (2013)	-	-	31.4

Table 5: F_1 on TEST with Gold Retrieval and Freebase Search API as the IR front end. Berant et al. (2013) actually reported *accuracy* on this dataset. However, since their system predicted answers for almost every question (p.c.), it is roughly that precision=recall=F_1=accuracy for them.

features improved overall F_1 by 5%, a 13% relative improvement: a remarkable gain given that the model already learned a strong correlation between question types and answer types (explained more in discussion and Table 6 later).

Finally, the ratio of positive vs. negative examples affect final F_1: the more positive examples, the lower the precision and the higher the recall. Under the original setting, this ratio was about $1:275$. This produced precision around 60% and recall around 35% (c.f. Table 4). To optimize for F_1, we down-sampled the negative examples to 20%, i.e., a new ratio of $1:55$. This boosted the final F_1 on DEV to 48%. We report the final TEST result under this down-sampled training. In practice the precision/recall balance can be adjusted by the positive/negative ratio.

Test Results Table 5 gives the final F_1 on TEST. "Gold Retrieval" always ranked the correct topic node top 1, a perfect IR front-end assumption. In a more realistic scenario, we had already evaluated that the Freebase Search API returned the correct topic node 95% of the time in its top 10 results (c.f. Table 3), thus we also tested on the top 10 results returned by the Search API. To keep things simple, we did not perform answer voting, but simply extracted answers from the first (ranked by the Search API) topic node with predicted answer(s) found. The final F_1 of 42.0% gives a relative improvement over previous best result (Berant et al., 2013) of 31.4% by one third.

One question of interest is whether our system, aided by the massive web data, can be fairly compared to the semantic parsing approaches (note that Berant et al. (2013) also used ClueWeb indirectly through `ReVerb`). Thus we took out the word overlapping and CluewebMapping based features, and the new F_1 on TEST was 36.9%.

The other question of interest is that whether our system has acquired some level of "machine intelligence": how much does it know what the question inquires? We discuss it below through feature and error analysis.

wgt.	feature
5.56	qfocus=money|type=Currency
5.35	qverb=die|type=Cause_Of_Death
5.11	qword=when|type=datetime
4.56	qverb=border|rel=location.adjoins
3.90	qword=why|incoming_relation_rank=top_3
2.94	qverb=go|qtopic=location|type=Tourist_attraction
-3.94	qtopic=location|rel=location.imports_exports.date
-2.93	qtopic=person|rel=education.end_date

Table 6: A sample of the top 50 most positive/negative features. Features are production between question and node features (c.f. Figure 1).

Discussion The combination between questions and Freebase nodes captures some real gist of QA pattern typing, shown in Table 6 with sampled features and weights. Our system learned, for instance, when the question asks for geographic adjacency information (qverb=border), the correct answer relation to look for is location.adjoins. Detailed comparison with the output from Berant et al. (2013) is a work in progress and will be presented in a follow-up report.

7 Conclusion

We proposed an automatic method for Question Answering from structured data source (Freebase). Our approach associates question features with answer patterns described by Freebase and has achieved state-of-the-art results on a balanced and realistic QA corpus. To compensate for the problem of domain mismatch or overfitting, we exploited ClueWeb, mined mappings between KB relations and natural language text, and showed that it helped both relation prediction and answer extraction. Our method employs relatively lightweight machinery but has good performance. We hope that this result establishes a new baseline against which semantic parsing researchers can measure their progress towards deeper language understanding and answering of human questions.

Acknowledgments We thank the Allen Institute for Artificial Intelligence for funding this work. We are also grateful to Jonathan Berant, Tom Kwiatkowski, Qingqing Cai, Adam Lopez, Chris Callison-Burch and Peter Clark for helpful discussion and to the reviewers for insightful comments.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBPedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. In *Proceedings of EMNLP*.

Steven Bird and Edward Loper. 2004. NLTK: The Natural Language Toolkit. In *Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM.

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

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of ACL*.

David L Chen and Raymond J Mooney. 2011. Learning to Interpret Natural Language Navigation Instructions from Observations. In *AAAI*, volume 2, pages 1–2.

J. Chu-Carroll, J. Fan, B. K. Boguraev, D. Carmel, D. Sheinwald, and C. Welty. 2012. Finding needles in the haystack: Search and candidate generation. *IBM Journal of Research and Development*.

Jérôme Euzenat and Pavel Shvaiko. 2007. *Ontology matching*. Springer.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of EMNLP*.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-Driven Learning for Open Question Answering. In *Proceedings of ACL*.

Anette Frank, Hans-Ulrich Krieger, Feiyu Xu, Hans Uszkoreit, Berthold Crysmann, Brigitte Jörg, and Ulrich Schäfer. 2007. Question answering from structured knowledge sources. *Journal of Applied Logic*, 5(1):20–48.

Freebase. 2013a. Freebase Search API. https://developers.google.com/freebase/v1/search-overview.

Freebase. 2013b. Freebase Topic API. https://developers.google.com/freebase/v1/topic-overview.

Evgeniy Gabrilovich, Michael Ringgaard, , and Amarnag Subramanya. 2013. FACC1: Freebase annotation of ClueWeb corpora, Version 1 (Release date 2013-06-26, Format version 1, Correction level 0). http://lemurproject.org/clueweb09/FACC1/, June.

Bert F Green Jr, Alice K Wolf, Carol Chomsky, and Kenneth Laughery. 1961. Baseball: an automatic question-answerer. In *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, pages 219–224. ACM.

Jerry R Hobbs. 1985. Ontological promiscuity. In *Proceedings of ACL*.

Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, Edwin Lewis-Kelham, Gerard De Melo, and Gerhard Weikum. 2011. Yago2: exploring and querying world knowledge in time, space, context, and many languages. In *Proceedings of the 20th international conference companion on World Wide Web*, pages 229–232. ACM.

Bevan Keeley Jones, Mark Johnson, and Sharon Goldwater. 2012. Semantic parsing with bayesian tree transducers. In *Proceedings of ACL*.

Rohit J Kate and Raymond J Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of ACL*.

Tibor Kiss and Jan Strunk. 2006. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525.

Philipp Koehn. 2010. *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA.

Jayant Krishnamurthy and Tom M Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of EMNLP-CoNLL*.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of EMNLP*, pages 1223–1233.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of EMNLP*.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling Semantic Parsers with On-the-fly Ontology Matching. In *Proceedings of EMNLP*.

Percy Liang, Michael I. Jordan, and Dan Klein. 2011. Learning Dependency-Based Compositional Semantics. In *Proceedings of ACL*.

Thomas Lin, Oren Etzioni, et al. 2012. Entity Linking at Web Scale. In *Proceedings of Knowledge Extraction Workshop (AKBC-WEKEX)*, pages 84–88.

Vanessa Lopez, Michele Pasin, and Enrico Motta. 2005. Aqualog: An ontology-portable question answering system for the semantic web. In *The Semantic Web: Research and Applications*, pages 546–562. Springer.

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

Naoaki Okazaki. 2009. Classias: a collection of machine-learning algorithms for classification.

Dave Orr, Amar Subramanya, Evgeniy Gabrilovich, and Michael Ringgaard. 2013. 11 billion clues in 800 million documents: A web research corpus annotated with freebase concepts. http://googleresearch.blogspot.com/2013/07/11-billion-clues-in-800-million.html, July.

Erhard Rahm and Philip A Bernstein. 2001. A survey of approaches to automatic schema matching. *the VLDB Journal*, 10(4):334–350.

Saeedeh Shekarpour, Axel-Cyrille Ngonga Ngomo, and Sören Auer. 2013. Question answering on interlinked data. In *Proceedings of WWW*.

Lappoon R Tang and Raymond J Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Machine Learning: ECML 2001*, pages 466–477. Springer.

Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. 2012. Template-based question answering over RDF data. In *Proceedings of the 21st international conference on World Wide Web*.

Yuk Wah Wong and Raymond J Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of ACL*.

William A Woods. 1977. Lunar rocks in natural english: Explorations in natural language question answering. *Linguistic structures processing*, 5:521–569.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, Maya Ramanath, Volker Tresp, and Gerhard Weikum. 2012. Natural language questions for the web of data. In *Proceedings of EMNLP*.

Luke S Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *Uncertainty in Artificial Intelligence (UAI)*.

Luke S Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of EMNLP-CoNLL*.

Luke S Zettlemoyer and Michael Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proceedings of ACL-CoNLL*.

Knowledge-Based Question Answering as Machine Translation

Junwei Bao[†] *, **Nan Duan**[‡] , **Ming Zhou**[‡] , **Tiejun Zhao**[†]
[†]Harbin Institute of Technology
[‡]Microsoft Research
baojunwei001@gmail.com
{nanduan, mingzhou}@microsoft.com
tjzhao@hit.edu.cn

Abstract

A typical knowledge-based question answering (KB-QA) system faces two challenges: one is to transform natural language questions into their meaning representations (MRs); the other is to retrieve answers from knowledge bases (KBs) using generated MRs. Unlike previous methods which treat them in a cascaded manner, we present a translation-based approach to solve these two tasks in one unified framework. We translate questions to answers based on CYK parsing. Answers as translations of the span covered by each CYK cell are obtained by a question translation method, which first generates formal triple queries as MRs for the span based on question patterns and relation expressions, and then retrieves answers from a given KB based on triple queries generated. A linear model is defined over derivations, and minimum error rate training is used to tune feature weights based on a set of question-answer pairs. Compared to a KB-QA system using a state-of-the-art semantic parser, our method achieves better results.

1 Introduction

Knowledge-based question answering (KB-QA) computes answers to natural language (NL) questions based on existing knowledge bases (KBs). Most previous systems tackle this task in a cascaded manner: First, the input question is transformed into its meaning representation (MR) by an independent semantic parser (Zettlemoyer and Collins, 2005; Mooney, 2007; Artzi and Zettlemoyer, 2011; Liang et al., 2011; Cai and Yates, 2013; Poon, 2013; Artzi et al., 2013; Kwiatkowski et al., 2013; Berant et al., 2013); Then, the answers are retrieved from existing KBs using generated MRs as queries.

Unlike existing KB-QA systems which treat semantic parsing and answer retrieval as two cascaded tasks, this paper presents a unified framework that can integrate semantic parsing into the question answering procedure directly. Borrowing ideas from machine translation (MT), we treat the QA task as a translation procedure. Like MT, CYK parsing is used to parse each input question, and answers of the span covered by each CYK cell are considered the translations of that cell; unlike MT, which uses offline-generated translation tables to translate source phrases into target translations, a semantic parsing-based question translation method is used to translate each span into its answers on-the-fly, based on question patterns and relation expressions. The final answers can be obtained from the root cell. Derivations generated during such a translation procedure are modeled by a linear model, and minimum error rate training (MERT) (Och, 2003) is used to tune feature weights based on a set of question-answer pairs.

Figure 1 shows an example: the question *director of movie starred by Tom Hanks* is translated to one of its answers *Robert Zemeckis* by three main steps: (i) translate *director of* to *director of*; (ii) translate *movie starred by Tom Hanks* to one of its answers *Forrest Gump*; (iii) translate *director of Forrest Gump* to a final answer *Robert Zemeckis*. Note that the updated question covered by Cell[0, 6] is obtained by combining the answers to question spans covered by Cell[0, 1] and Cell[2, 6].

The contributions of this work are two-fold: (1) We propose a translation-based KB-QA method that integrates semantic parsing and QA in one unified framework. The benefit of our method is that we don't need to explicitly generate complete semantic structures for input questions. Be-

*This work was finished while the author was visiting Microsoft Research Asia.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 967–976,
Baltimore, Maryland, USA, June 23-25 2014.

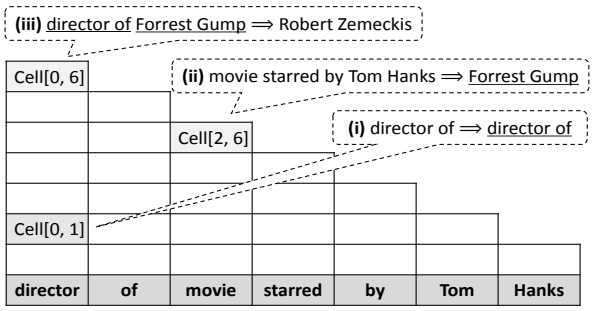


Figure 1: Translation-based KB-QA example

sides which, answers generated during the translation procedure help significantly with search space pruning. (2) We propose a robust method to transform single-relation questions into formal triple queries as their MRs, which trades off between transformation accuracy and recall using question patterns and relation expressions respectively.

2 Translation-Based KB-QA

2.1 Overview

Formally, given a knowledge base $\mathcal{KB}$ and an N-L question $\mathcal{Q}$, our KB-QA method generates a set of formal triples-answer pairs $\{\langle \mathcal{D}, \mathcal{A} \rangle\}$ as derivations, which are scored and ranked by the distribution $P(\langle \mathcal{D}, \mathcal{A} \rangle | \mathcal{KB}, \mathcal{Q})$ defined as follows:

$$\frac{exp\{\sum_{i=1}^{M} \lambda_i \cdot h_i(\langle \mathcal{D}, \mathcal{A} \rangle, \mathcal{KB}, \mathcal{Q})\}}{\sum_{\langle \mathcal{D}', \mathcal{A}' \rangle \in \mathcal{H}(\mathcal{Q})} exp\{\sum_{i=1}^{M} \lambda_i \cdot h_i(\langle \mathcal{D}', \mathcal{A}' \rangle, \mathcal{KB}, \mathcal{Q})\}}$$

- $\mathcal{KB}$ denotes a knowledge base[1] that stores a set of assertions. Each assertion $t \in \mathcal{KB}$ is in the form of $\{e_{sbj}^{ID}, p, e_{obj}^{ID}\}$, where p denotes a predicate, e_{sbj}^{ID} and e_{obj}^{ID} denote the subject and object entities of t, with unique IDs[2].

- $\mathcal{H}(\mathcal{Q})$ denotes the search space $\{\langle \mathcal{D}, \mathcal{A} \rangle\}$. $\mathcal{D}$ is composed of a set of *ordered* formal triples $\{t_1, ..., t_n\}$. Each triple $t = \{e_{sbj}, p, e_{obj}\}_i^j \in \mathcal{D}$ denotes an assertion in $\mathcal{KB}$, where i and j denotes the beginning and end indexes of the question span from which t is transformed. The order of triples in $\mathcal{D}$ denotes the order of translation steps from $\mathcal{Q}$ to $\mathcal{A}$. E.g., $\langle$*director of, Null, director of*$\rangle_0^1$, $\langle$*Tom Hanks, Film.Actor.Film, Forrest Gump*$\rangle_2^6$ and $\langle$*Forrest Gump, Film.Film.Director, Robert Zemeckis*$\rangle_0^6$ are three ordered formal triples corresponding to the three translation steps in Figure 1. We define the task of transforming question spans into formal triples as *question translation*. $\mathcal{A}$ denotes one final answer of $\mathcal{Q}$.

- $h_i(\cdot)$ denotes the i^{th} feature function.

- λ_i denotes the feature weight of $h_i(\cdot)$.

According to the above description, our KB-QA method can be decomposed into four tasks as: (1) search space generation for $\mathcal{H}(\mathcal{Q})$; (2) question translation for transforming question spans into their corresponding formal triples; (3) feature design for $h_i(\cdot)$; and (4) feature weight tuning for $\{\lambda_i\}$. We present details of these four tasks in the following subsections one-by-one.

2.2 Search Space Generation

We first present our translation-based KB-QA method in Algorithm 1, which is used to generate $\mathcal{H}(\mathcal{Q})$ for each input NL question $\mathcal{Q}$.

Algorithm 1: Translation-based KB-QA

```
for l = 1 to |Q| do
    for all i, j s.t. j − i = l do
        H(Q_i^j) = ∅;
        T = QTrans(Q_i^j, KB);
        foreach formal triple t ∈ T do
            create a new derivation d;
            d.A = t.e_obj;
            d.D = {t};
            update the model score of d;
            insert d to H(Q_i^j);
        end
    end
end
for l = 1 to |Q| do
    for all i, j s.t. j − i = l do
        for all m s.t. i ≤ m < j do
            for d_l ∈ H(Q_i^m) and d_r ∈ H(Q_{m+1}^j) do
                Q_update = d_l.A + d_r.A;
                T = QTrans(Q_update, KB);
                foreach formal triple t ∈ T do
                    create a new derivation d;
                    d.A = t.e_obj;
                    d.D = d_l.D ⋃ d_r.D ⋃ {t};
                    update the model score of d;
                    insert d to H(Q_i^j);
                end
            end
        end
    end
end
return H(Q).
```

[1] We use a large scale knowledge base in this paper, which contains 2.3B entities, 5.5K predicates, and 18B assertions. A 16-machine cluster is used to host and serve the whole data.

[2] Each KB entity has a unique ID. For the sake of convenience, we omit the ID information in the rest of the paper.

The first half (from Line 1 to Line 13) generates a formal triple set T for each unary span $\mathcal{Q}_i^j \in \mathcal{Q}$, using the question translation method $QTrans(\mathcal{Q}_i^j, \mathcal{KB})$ (Line 4), which takes $\mathcal{Q}_i^j$ as the input. Each triple $t \in T$ returned is in the form of $\{e_{sbj}, p, e_{obj}\}$, where e_{sbj}'s mention occurs in $\mathcal{Q}_i^j$, p is a predicate that denotes the meaning expressed by the context of e_{sbj} in $\mathcal{Q}_i^j$, e_{obj} is an answer of $\mathcal{Q}_i^j$ based on e_{sbj}, p and $\mathcal{KB}$. We describe the implementation detail of $QTrans(\cdot)$ in Section 2.3.

The second half (from Line 14 to Line 31) first updates the content of each bigger span $\mathcal{Q}_i^j$ by concatenating the answers to its any two consecutive smaller spans covered by $\mathcal{Q}_i^j$ (Line 18). Then, $QTrans(\mathcal{Q}_i^j, \mathcal{KB})$ is called to generate triples for the updated span (Line 19). The above operations are equivalent to answering a simplified question, which is obtained by replacing the answerable spans in the original question with their corresponding answers. The search space $\mathcal{H}(\mathcal{Q})$ for the entire question $\mathcal{Q}$ is returned at last (Line 31).

2.3 Question Translation

The purpose of question translation is to translate a span $\mathcal{Q}$ to a set of formal triples T. Each triple $t \in T$ is in the form of $\{e_{sbj}, p, e_{obj}\}$, where e_{sbj}'s mention[3] occurs in $\mathcal{Q}$, p is a predicate that denotes the meaning expressed by the context of e_{sbj} in $\mathcal{Q}$, e_{obj} is an answer to $\mathcal{Q}$ retrieved from $\mathcal{KB}$ using a triple query $q = \{e_{sbj}, p, ?\}$. Note that if no predicate p or answer e_{obj} can be generated, $\{\mathcal{Q}, Null, \mathcal{Q}\}$ will be returned as a special triple, which sets e_{obj} to be $\mathcal{Q}$ itself, and p to be $Null$. This makes sure the un-answerable spans can be passed on to the higher-level operations.

Question translation assumes each span $\mathcal{Q}$ is a *single-relation question* (Fader et al., 2013). Such assumption simplifies the efforts of semantic parsing to the minimum question units, while leaving the capability of handling multiple-relation questions (Figure 1 gives one such example) to the outer CYK-parsing based translation procedure. Two question translation methods are presented in the rest of this subsection, which are based on question patterns and relation expressions respectively.

2.3.1 Question Pattern-based Translation

A question pattern $\mathcal{QP}$ includes a pattern string $\mathcal{QP}_{pattern}$, which is composed of words and a slot symbol $[Slot]$, and a KB predicate $\mathcal{QP}_{predicate}$, which denotes the meaning expressed by the context words in $\mathcal{QP}_{pattern}$.

Algorithm 2: $\mathcal{QP}$-based Question Translation

```
1  T = ∅;
2  foreach entity mention e_Q ∈ Q do
3      Q_pattern = replace e_Q in Q with [Slot];
4      foreach question pattern QP do
5          if Q_pattern == QP_pattern then
6              E = Disambiguate(e_Q, QP_predicate);
7              foreach e ∈ E do
8                  create a new triple query q;
9                  q = {e, QP_predicate, ?};
10                 {A_i} = AnswerRetrieve(q, KB);
11                 foreach A ∈ {A_i} do
12                     create a new formal triple t;
13                     t = {q.e_sbj, q.p, A};
14                     t.score = 1.0;
15                     insert t to T;
16                 end
17             end
18         end
19     end
20 end
21 return T.
```

Algorithm 2 shows how to generate formal triples for a span $\mathcal{Q}$ based on question patterns ($\mathcal{QP}$-based question translation). For each entity mention $e_{\mathcal{Q}} \in \mathcal{Q}$, we replace it with $[Slot]$ and obtain a pattern string $\mathcal{Q}_{pattern}$ (Line 3). If $\mathcal{Q}_{pattern}$ can match one $\mathcal{QP}_{pattern}$, then we construct a triple query q (Line 9) using $\mathcal{QP}_{predicate}$ as its predicate and one of the KB entities returned by $Disambiguate(e_{\mathcal{Q}}, \mathcal{QP}_{predicate})$ as its subject entity (Line 6). Here, the objective of $Disambiguate(e_{\mathcal{Q}}, \mathcal{QP}_{predicate})$ is to output a set of disambiguated KB entities $\mathcal{E}$ in $\mathcal{KB}$. The name of each entity returned equals the input entity mention $e_{\mathcal{Q}}$ and occurs in some assertions where $\mathcal{QP}_{predicate}$ are the predicates. The underlying idea is to use the context (predicate) information to help entity disambiguation. The answers of q are returned by $AnswerRetrieve(q, \mathcal{KB})$ based on q and $\mathcal{KB}$ (Line 10), each of which is used to construct a formal triple and added to T for $\mathcal{Q}$ (from Line 11 to Line 16). Figure 2 gives an example.

Question patterns are collected as follows: First, *5W* queries, which begin with What, Where, Who, When, or Which, are selected from a large scale query log of a commercial search engine; Then, a cleaned entity dictionary is used to annotate each query by replacing all entity mentions it contains with the symbol $[Slot]$. Only high-frequent query patterns which contain one $[Slot]$ are maintained;

[3]For simplicity, a cleaned entity dictionary dumped from the entire $\mathcal{KB}$ is used to detect entity mentions in $\mathcal{Q}$.

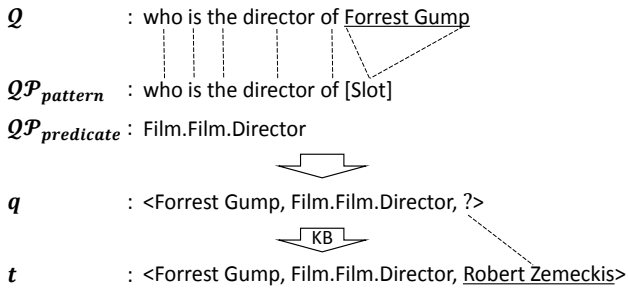


Figure 2: $\mathcal{QP}$-based question translation example

Lastly, annotators try to manually label the most-frequent 50,000 query patterns with their corresponding predicates, and 4,764 question patterns with single labeled predicates are obtained.

From experiments (Table 3 in Section 4.3) we can see that, question pattern based question translation can achieve high end-to-end accuracy. But as human efforts are needed in the mining procedure, this method cannot be extended to large scale very easily. Besides, different users often type the questions with the same meaning in different NL expressions. For example, although the question *Forrest Gump was directed by which moviemaker* means the same as the question $\mathcal{Q}$ in Figure 2, no question pattern can cover it. We need to find an alternative way to alleviate such coverage issue.

2.3.2 Relation Expression-based Translation

Aiming to alleviate the coverage issue occurring in $\mathcal{QP}$-based method, an alternative relation expression ($\mathcal{RE}$) -based method is proposed, and will be used when the $\mathcal{QP}$-based method fails.

We define $\mathcal{RE}_p$ as a relation expression set for a given KB predicate $p \in \mathcal{KB}$. Each relation expression $\mathcal{RE} \in \mathcal{RE}_p$ includes an expression string $\mathcal{RE}_{expression}$, which must contain at least one content word, and a weight $\mathcal{RE}_{weight}$, which denotes the confidence that $\mathcal{RE}_{expression}$ can represent p's meaning in NL. For example, *is the director of* is one relation expression string for the predicate *Film.Film.Director*, which means it is usually used to express this relation (predicate) in NL.

Algorithm 3 shows how to generate triples for a question $\mathcal{Q}$ based on relation expressions. For each possible entity mention $e_{\mathcal{Q}} \in \mathcal{Q}$ and a KB predicate $p \in \mathcal{KB}$ that is related to a KB entity e whose name equals $e_{\mathcal{Q}}$, $Sim(e_{\mathcal{Q}}, \mathcal{Q}, \mathcal{RE}_p)$ is computed (Line 5) based on the similarity between question context and $\mathcal{RE}_p$, which measures how likely $\mathcal{Q}$ can be transformed into a triple query

Algorithm 3: $\mathcal{RE}$-based Question Translation

```
1  T = ∅;
2  foreach entity mention e_Q ∈ Q do
3      foreach e ∈ KB s.t. e.name==e_Q do
4          foreach predicate p ∈ KB related to e do
5              score = Sim(e_Q, Q, RE_p);
6              if score > 0 then
7                  create a new triple query q;
8                  q = {e, p, ?};
9                  {A_i} = AnswerRetrieve(q, KB);
10                 foreach A ∈ {A_i} do
11                     create a new formal triple t;
12                     t = {q.e_sbj, q.p, A};
13                     t.score = score;
14                     insert t to T;
15                 end
16             end
17         end
18     end
19 end
20 sort T based on the score of each t ∈ T;
21 return T.
```

$q = \{e, p, ?\}$. If this score is larger than 0, which means there are overlaps between $\mathcal{Q}$'s context and $\mathcal{RE}_p$, then q will be used as the triple query of $\mathcal{Q}$, and a set of formal triples will be generated based on q and $\mathcal{KB}$ (from Line 7 to Line 15). The computation of $Sim(e_{\mathcal{Q}}, \mathcal{Q}, \mathcal{RE}_p)$ is defined as follows:

$$\sum_{n} \frac{1}{|\mathcal{Q}| - n + 1} \cdot \{ \sum_{\omega_n \in \mathcal{Q}, \omega_n \bigcap e_{\mathcal{Q}} = \phi} P(\omega_n | \mathcal{RE}_p) \}$$

where n is the n-gram order which ranges from 1 to 5, ω_n is an n-gram occurring in $\mathcal{Q}$ without overlapping with $e_{\mathcal{Q}}$ and containing at least one content word, $P(\omega_n|\mathcal{RE}_p)$ is the posterior probability which is computed by:

$$P(\omega_n|\mathcal{RE}_p) = \frac{Count(\omega_n, \mathcal{RE}_p)}{\sum_{\omega'_n \in \mathcal{RE}_p} Count(\omega'_n, \mathcal{RE}_p)}$$

$Count(\omega, \mathcal{RE}_p)$ denotes the weighted sum of times that ω occurs in $\mathcal{RE}_p$:

$$Count(\omega, \mathcal{RE}_p) = \sum_{\mathcal{RE} \in \mathcal{RE}_p} \{\#_{\omega}(\mathcal{RE}) \cdot \mathcal{RE}_{weight}\}$$

where $\#_{\omega}(\mathcal{RE})$ denotes the number of times that ω occurs in $\mathcal{RE}_{expression}$, and $\mathcal{RE}_{weight}$ is decided by the relation expression extraction component.

Figure 3 gives an example, where n-grams with rectangles are the ones that occur in both $\mathcal{Q}$'s context and the relation expression set of a given predicate $p = Film.Film.Director$. Unlike the $\mathcal{QP}$-based method which needs a perfect match, the

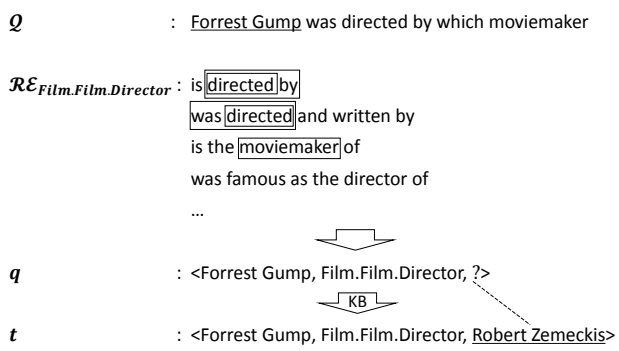


Figure 3: $\mathcal{RE}$-based question translation example

$\mathcal{RE}$-based method allows fuzzy matching between $\mathcal{Q}$ and $\mathcal{RE}_p$, and records this (Line 13) in generated triples, which is used as features later.

Relation expressions are mined as follows: Given a set of KB assertions with an identical predicate p, we first extract all sentences from English Wiki pages[4], each of which contains at least one pair of entities occurring in one assertion. Then, we extract the shortest path between paired entities in the dependency tree of each sentence as an $\mathcal{RE}$ candidate for the given predicate. The intuition is that any sentence containing such entity pairs occur in an assertion is likely to express the predicate of that assertion in some way. Last, all relation expressions extracted are filtered by heuristic rules, i.e., the frequency must be larger than 4, the length must be shorter than 10, and then weighted by the pattern scoring methods proposed in (Gerber and Ngomo, 2011; Gerber and Ngomo, 2012). For each predicate, we only keep the relation expressions whose pattern scores are larger than a pre-defined threshold. Figure 4 gives one relation expression extraction example. The statistics and overall quality of the relation expressions are listed in Section 4.1.

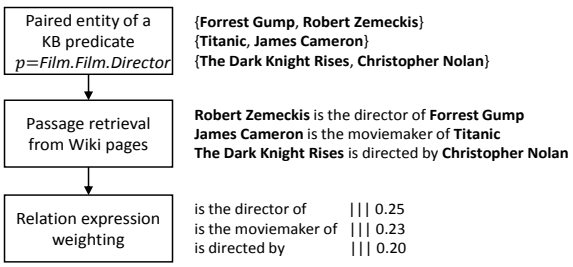


Figure 4: $\mathcal{RE}$ extraction example

[4]http://en.wikipedia.org/wiki/Wikipedia:Database_download

2.3.3 Question Decomposition

Sometimes, a question may provide multiple constraints to its answers. *movie starred by Tom Hanks in 1994* is one such question. All the films as the answers of this question should satisfy the following two constraints: (1) starred *by Tom Hanks*; and (2) released *in 1994*. It is easy to see that such questions cannot be translated to single triples.

We propose a dependency tree-based method to handle such multiple-constraint questions by (i) decomposing the original question into a set of *sub-questions* using syntax-based patterns; and (ii) intersecting the answers of all sub-questions as the final answers of the original question. Note, question decomposition only operates on the original question and question spans covered by complete dependency subtrees. Four syntax-based patterns (Figure 5) are used for question decomposition. If a question matches any one of these patterns, then sub-questions are generated by collecting the paths between n_0 and each $n_i(i > 0)$ in the pattern, where each n denotes a complete subtree with a noun, number, or question word as its root node, the symbol $*$ above $prep^*$ denotes this preposition can be skipped in matching. For the question mentioned at the beginning, its two sub-questions generated are *movie starred by Tom Hanks* and *movie starred in 1994*, as its dependency form matches pattern (a). Similar ideas are used in IBM Watson (Kalyanpur et al., 2012) as well.

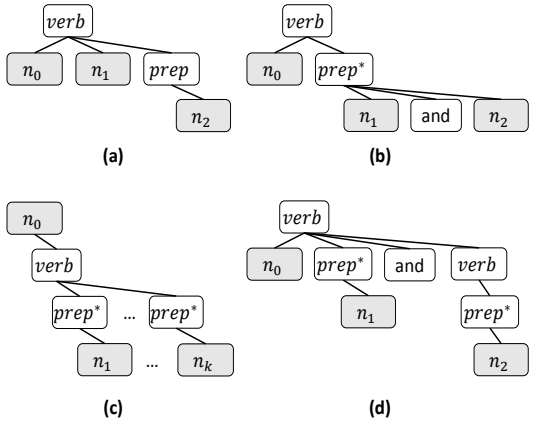


Figure 5: Four syntax-based patterns for question decomposition

As dependency parsing is not perfect, we generate single triples for such questions without considering constraints as well, and add them to the search space for competition. $h_{syntax\_constraint}(\cdot)$

is used to boost triples that are converted from sub-questions generated by question decomposition. The more constraints an answer satisfies, the better. Obviously, current patterns used can't cover all cases but most-common ones. We leave a more general pattern mining method for future work.

2.4 Feature Design

The objective of our KB-QA system is to seek the derivation $\langle \hat{\mathcal{D}}, \hat{\mathcal{A}} \rangle$ that maximizes the probability $P(\langle \mathcal{D}, \mathcal{A} \rangle | \mathcal{KB}, \mathcal{Q})$ described in Section 2.1 as:

$$\begin{aligned} \langle \hat{\mathcal{D}}, \hat{\mathcal{A}} \rangle &= \underset{\langle \mathcal{D}, \mathcal{A} \rangle \in \mathcal{H}(\mathcal{Q})}{argmax} P(\langle \mathcal{D}, \mathcal{A} \rangle | \mathcal{KB}, \mathcal{Q}) \\ &= \underset{\langle \mathcal{D}, \mathcal{A} \rangle \in \mathcal{H}(\mathcal{Q})}{argmax} \sum_{i=1}^{M} \lambda_i \cdot h_i(\langle \mathcal{D}, \mathcal{A} \rangle, \mathcal{KB}, \mathcal{Q}) \end{aligned}$$

We now introduce the feature sets $\{h_i(\cdot)\}$ that are used in the above linear model:

- $h_{question\_word}(\cdot)$, which counts the number of original question words occurring in $\mathcal{A}$. It penalizes those partially answered questions.
- $h_{span}(\cdot)$, which counts the number of spans in $\mathcal{Q}$ that are converted to formal triples. It controls the granularity of the spans used in question translation.
- $h_{syntax\_subtree}(\cdot)$, which counts the number of spans in $\mathcal{Q}$ that are (1) converted to formal triples, whose predicates are not $Null$, and (2) covered by complete dependency subtrees at the same time. The underlying intuition is that, dependency subtrees of $\mathcal{Q}$ should be treated as units for question translation.
- $h_{syntax\_constraint}(\cdot)$, which counts the number of triples in $\mathcal{D}$ that are converted from sub-questions generated by the question decomposition component.
- $h_{triple}(\cdot)$, which counts the number of triples in $\mathcal{D}$, whose predicates are not $Null$.
- $h_{triple_{weight}}(\cdot)$, which sums the scores of all triples $\{t_i\}$ in $\mathcal{D}$ as $\sum_{t_i \in \mathcal{D}} t_i.score$.
- $h_{\mathcal{QP}_{count}}(\cdot)$, which counts the number of triples in $\mathcal{D}$ that are generated by $\mathcal{QP}$-based question translation method.
- $h_{\mathcal{RE}_{count}}(\cdot)$, which counts the number of triples in $\mathcal{D}$ that are generated by $\mathcal{RE}$-based question translation method.
- $h_{staticrank_{sbj}}(\cdot)$, which sums the static rank scores of all subject entities in $\mathcal{D}$'s triple set as $\sum_{t_i \in \mathcal{D}} t_i.e_{sbj}.static\_rank$.
- $h_{staticrank_{obj}}(\cdot)$, which sums the static rank scores of all object entities in $\mathcal{D}$'s triple set as $\sum_{t_i \in \mathcal{D}} t_i.e_{obj}.static\_rank$.
- $h_{confidence_{obj}}(\cdot)$, which sums the confidence scores of all object entities in $\mathcal{D}$'s triple set as $\sum_{t \in \mathcal{D}} t.e_{obj}.confidence$.

For each assertion $\{e_{sbj}, p, e_{obj}\}$ stored in $\mathcal{KB}$, $e_{sbj}.static\_rank$ and $e_{obj}.static\_rank$ denote the static rank scores[5] for e_{sbj} and e_{obj} respectively; $e_{obj}.confidence\_rank$ represents the probability $p(e_{obj}|e_{sbj}, p)$. These three scores are used as features to rank answers generated in QA procedure.

2.5 Feature Weight Tuning

Given a set of question-answer pairs $\{\mathcal{Q}_i, \mathcal{A}_i^{ref}\}$ as the development (dev) set, we use the minimum error rate training (MERT) (Och, 2003) algorithm to tune the feature weights λ_i^M in our proposed model. The training criterion is to seek the feature weights that can minimize the accumulated errors of the top-1 answer of questions in the dev set:

$$\hat{\lambda}_1^M = \underset{\lambda_1^M}{argmin} \sum_{i=1}^{N} Err(\mathcal{A}_i^{ref}, \hat{\mathcal{A}}_i; \lambda_1^M)$$

N is the number of questions in the dev set, $\mathcal{A}_i^{ref}$ is the correct answers as references of the i^{th} question in the dev set, $\hat{\mathcal{A}}_i$ is the top-1 answer candidate of the i^{th} question in the dev set based on feature weights λ_1^M, $Err(\cdot)$ is the error function which is defined as:

$$Err(\mathcal{A}_i^{ref}, \hat{\mathcal{A}}_i; \lambda_1^M) = 1 - \delta(\mathcal{A}_i^{ref}, \hat{\mathcal{A}}_i)$$

where $\delta(\mathcal{A}_i^{ref}, \hat{\mathcal{A}}_i)$ is an indicator function which equals 1 when $\hat{\mathcal{A}}_i$ is included in the reference set $\mathcal{A}_i^{ref}$, and 0 otherwise.

3 Comparison with Previous Work

Our work intersects with two research directions: semantic parsing and question answering.

Some previous works on semantic parsing (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Wong and Mooney, 2006; Zettlemoyer and Collins, 2007; Wong and Mooney,

[5]The static rank score of an entity represents a general indicator of the overall quality of that entity.

2007; Kwiatkowski et al., 2010; Kwiatkowski et al., 2011) require manually annotated logical forms as supervision, and are hard to extend resulting parsers from limited domains, such as GEO, JOBS and ATIS, to open domains. Recent works (Clarke and Lapata, 2010; Liang et al., 2013) have alleviated such issues using question-answer pairs as weak supervision, but still with the shortcoming of using limited lexical triggers to link NL phrases to predicates. Poon (2013) has proposed an unsupervised method by adopting grounded-learning to leverage the database for indirect supervision. But transformation from NL questions to MRs heavily depends on dependency parsing results. Besides, the KB used (ATIS) is limited as well. Kwiatkowski et al. (2013) use Wiktionary and a limited manual lexicon to map POS tags to a set of predefined CCG lexical categories, which aims to reduce the need for learning lexicon from training data. But it still needs human efforts to define lexical categories, which usually can not cover all the semantic phenomena.

Berant et al. (2013) have not only enlarged the KB used for Freebase (Google, 2013), but also used a bigger lexicon trigger set extracted by the open IE method (Lin et al., 2012) for NL phrases to predicates linking. In comparison, our method has further advantages: (1) Question answering and semantic parsing are performed in an joint way under a unified framework; (2) A robust method is proposed to map NL questions to their formal triple queries, which trades off the mapping quality by using question patterns and relation expressions in a cascaded way; and (3) We use domain independent feature set which allowing us to use a relatively small number of question-answer pairs to tune model parameters.

Fader et al. (2013) map questions to formal (triple) queries over a large scale, open-domain database of facts extracted from a raw corpus by ReVerb (Fader et al., 2011). Compared to their work, our method gains an improvement in two aspects: (1) Instead of using facts extracted using the open IE method, we leverage a large scale, high-quality knowledge base; (2) We can handle multiple-relation questions, instead of single-relation queries only, based on our translation based KB-QA framework.

Espana-Bonet and Comas (2012) have proposed an MT-based method for factoid QA. But MT in there work means to translate questions into n-best translations, which are used for finding similar sentences in the document collection that probably contain answers. Echihabi and Marcu (2003) have developed a noisy-channel model for QA, which explains how a sentence containing an answer to a given question can be rewritten into that question through a sequence of stochastic operations. Compared to the above two MT-motivated QA work, our method uses MT methodology to translate questions to answers directly.

4 Experiment

4.1 Data Sets

Following Berant et al. (2013), we use the same subset of WEBQUESTIONS (3,778 questions) as the development set (Dev) for weight tuning in MERT, and use the other part of WEBQUESTIONS (2,032 questions) as the test set (Test). Table 1 shows the statistics of this data set.

Data Set	# Questions	# Words
WEBQUESTIONS	5,810	6.7

Table 1: Statistics of evaluation set. # Questions is the number of questions in a data set, # Words is the averaged word count of a question.

Table 2 shows the statistics of question patterns and relation expressions used in our KB-QA system. As all question patterns are collected with human involvement as we discussed in Section 2.3.1, the quality is very high (98%). We also sample 1,000 instances from the whole relation expression set and manually label their quality. The accuracy is around 89%. These two resources can cover 566 head predicates in our KB.

	# Entries	Accuracy
Question Patterns	4,764	98%
Relation Expressions	133,445	89%

Table 2: Statistics of question patterns and relation expressions.

4.2 KB-QA Systems

Since Berant et al. (2013) is one of the latest work which has reported QA results based on a large scale, general domain knowledge base (Freebase), we consider their evaluation result on WEBQUESTIONS as our baseline.

Our KB-QA system generates the k-best derivations for each question span, where k is set to 20.

The answers with the highest model scores are considered the best answers for evaluation. For evaluation, we follow Berant et al. (2013) to allow partial credit and score an answer using the F1 measure, comparing the predicted set of entities to the annotated set of entities.

One difference between these two systems is the KB used. Since Freebase is completely contained by our KB, we disallow all entities which are not included by Freebase. By doing so, our KB provides the same knowledge as Freebase does, which means we do not gain any extra advantage by using a larger KB. But we still allow ourselves to use the static rank scores and confidence scores of entities as features, as we described in Section 2.4.

4.3 Evaluation Results

We first show the overall evaluation results of our KB-QA system and compare them with baseline's results on Dev and Test. Note that we do not re-implement the baseline system, but just list their evaluation numbers reported in the paper. Comparison results are listed in Table 3.

	Dev (Accuracy)	Test (Accuracy)
Baseline	32.9%	31.4%
Our Method	42.5% **(+9.6%)**	37.5% **(+6.1%)**

Table 3: Accuracy on evaluation sets. $Accuracy$ is defined as the number of correctly answered questions divided by the total number of questions.

Table 3 shows our KB-QA method outperforms baseline on both Dev and Test. We think the potential reasons of this improvement include:

- Different methods are used to map NL *phrases* to KB predicates. Berant et al. (2013) have used a lexicon extracted from a subset of ReVerb triples (Lin et al., 2012), which is similar to the relation expression set used in question translation. But as our relation expressions are extracted by an in-house extractor, we can record their extraction-related statistics as extra information, and use them as features to measure the mapping quality. Besides, as a portion of entities in our KB are extracted from Wiki, we know the one-to-one correspondence between such entities and Wiki pages, and use this information in relation expression extraction for entity disambiguation. A lower disambiguation error rate results in better relation expressions.
- Question patterns are used to map NL *context* to KB predicates. Context can be either continuous or discontinues phrases. Although the size of this set is limited, they can actually cover head questions/queries[6] very well. The underlying intuition of using patterns is that those high-frequent questions/queries should and can be treated and solved in the QA task, by involving human effort at a relative small price but with very impressive accuracy.

In order to figure out the impacts of question patterns and relation expressions, another experiment (Table 4) is designed to evaluate their independent influences, where $\mathcal{QP}_{only}$ and $\mathcal{RE}_{only}$ denote the results of KB-QA systems which only allow question patterns and relation expressions in question translation respectively.

Settings	Test (Accuracy)	Test (Precision)
$\mathcal{QP}_{only}$	11.8%	97.5%
$\mathcal{RE}_{only}$	32.5%	73.2%

Table 4: Impacts of question patterns and relation expressions. $Precision$ is defined as the number of correctly answered questions divided by the number of questions with non-empty answers generated by our KB-QA system.

From Table 4 we can see that the accuracy of $\mathcal{RE}_{only}$ on Test (32.5%) is slightly better than baseline's result (31.4%). We think this improvement comes from two aspects: (1) The quality of the relation expressions is better than the quality of the lexicon entries used in the baseline; and (2) We use the extraction-related statistics of relation expressions as features, which brings more information to measure the confidence of mapping between NL phrases and KB predicates, and makes the model to be more flexible. Meanwhile, $\mathcal{QP}_{only}$ perform worse (11.8%) than $\mathcal{RE}_{only}$, due to coverage issue. But by comparing the precisions of these two settings, we find $\mathcal{QP}_{only}$ (97.5%) outperforms $\mathcal{RE}_{only}$ (73.2%) significantly, due to its high quality. This means how to extract high-quality question patterns is worth to be studied for the question answering task.

As the performance of our KB-QA system relies heavily on the k-best beam approximation, we evaluate the impact of the beam size and list the comparison results in Figure 6. We can see that as

[6]Head questions/queries mean the questions/queries with high frequency and clear patterns.

we increase k incrementally, the accuracy increase at the same time. However, a larger k (e.g. 200) cannot bring significant improvements comparing to a smaller one (e.g., 20), but using a large k has a tremendous impact on system efficiency. So we choose $k = 20$ as the optimal value in above experiments, which trades off between accuracy and efficiency.

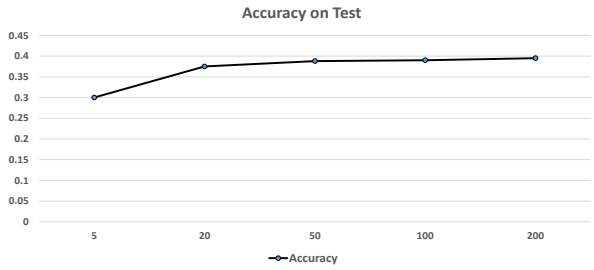


Figure 6: Impacts of beam size on accuracy.

Actually, the size of our system's search space is much smaller than the one of the semantic parser used in the baseline.This is due to the fact that, if triple queries generated by the question translation component cannot derive any answer from KB, we will discard such triple queries directly during the QA procedure. We can see that using a small k can achieve better results than baseline, where the beam size is set to be 200.

4.4 Error Analysis

4.4.1 Entity Detection

Since named entity recognizers trained on Penn TreeBank usually perform poorly on web queries, We instead use a simple string-match method to detect entity mentions in the question using a cleaned entity dictionary dumped from our KB. One problem of doing so is the entity detection issue. For example, in the question *who was Esther's husband ?*, we cannot detect *Esther* as an entity, as it is just part of an entity name. We need an ad-hoc entity detection component to handle such issues, especially for a web scenario, where users often type entity names in their partial or abbreviation forms.

4.4.2 Predicate Mapping

Some questions lack sufficient evidences to detect predicates. *where is Byron Nelson 2012 ?* is an example. Since each relation expression must contain at least one content word, this question cannot match any relation expression. Except for *Byron Nelson* and *2012*, all the others are non-content words.

Besides, ambiguous entries contained in relation expression sets of different predicates can bring mapping errors as well. For the following question *who did Steve Spurrier play pro football for?* as an example, since the unigram *play* exists in both *Film.Film.Actor* and *American_Football.Player.Current_Team* 's relation expression sets, we made a wrong prediction, which led to wrong answers.

4.4.3 Specific Questions

Sometimes, we cannot give exact answers to superlative questions like *what is the first book Sherlock Holmes appeared in?*. For this example, we can give all book names where *Sherlock Holmes* appeared in, but we cannot rank them based on their publication date , as we cannot learn the alignment between the constraint word *first* occurred in the question and the predicate *Book.Written_Work.Date_Of_First_Publication* from training data automatically. Although we have followed some work (Poon, 2013; Liang et al., 2013) to handle such special linguistic phenomena by defining some specific operators, it is still hard to cover all unseen cases. We leave this to future work as an independent topic.

5 Conclusion and Future Work

This paper presents a translation-based KB-QA method that integrates semantic parsing and QA in one unified framework. Comparing to the baseline system using an independent semantic parser with state-of-the-art performance, we achieve better results on a general domain evaluation set.

Several directions can be further explored in the future: (i) We plan to design a method that can extract question patterns automatically, using existing labeled question patterns and KB as weak supervision. As we discussed in the experiment part, how to mine high-quality question patterns is worth further study for the QA task; (ii) We plan to integrate an ad-hoc NER into our KB-QA system to alleviate the entity detection issue; (iii) In fact, our proposed QA framework can be generalized to other intelligence besides knowledge bases as well. Any method that can generate answers to questions, such as the Web-based QA approach, can be integrated into this framework, by using them in the question translation component.

References

Yoav Artzi and Luke S. Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *EMNLP*, pages 421–432.

Yoav Artzi, Nicholas FitzGerald, and Luke S. Zettlemoyer. 2013. Semantic parsing with combinatory categorial grammars. In *ACL (Tutorial Abstracts)*, page 2.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*, pages 1533–1544.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *ACL*, pages 423–433.

James Clarke and Mirella Lapata. 2010. Discourse constraints for document compression. *Computational Linguistics*, 36(3):411–441.

Abdessamad Echihabi and Daniel Marcu. 2003. A noisy-channel approach to question answering. In *ACL*.

Cristina Espana-Bonet and Pere R. Comas. 2012. Full machine translation for factoid question answering. In *EACL*, pages 20–29.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *EMNLP*, pages 1535–1545.

Anthony Fader, Luke S. Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *ACL*, pages 1608–1618.

Daniel Gerber and Axel-Cyrille Ngonga Ngomo. 2011. Bootstrapping the linked data web. In *ISWC*.

Daniel Gerber and Axel-Cyrille Ngonga Ngomo. 2012. Extracting multilingual natural-language patterns for rdf predicates. In *ESWC*.

Google. 2013. Freebase. In *http://www.freebase.com*.

Aditya Kalyanpur, Siddharth Patwardhan, Branimir Boguraev, Adam Lally, and Jennifer Chu-Carroll. 2012. Fact-based question decomposition in deepqa. *IBM Journal of Research and Development*, 56(3):13.

Tom Kwiatkowski, Luke S. Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic ccg grammars from logical form with higher-order unification. In *EMNLP*, pages 1223–1233.

Tom Kwiatkowski, Luke S. Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in ccg grammar induction for semantic parsing. In *EMNLP*, pages 1512–1523.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke S. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *EMNLP*, pages 1545–1556.

Percy Liang, Michael I. Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *ACL*, pages 590–599.

Percy Liang, Michael I. Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.

Thomas Lin, Mausam, and Oren Etzioni. 2012. Entity linking at web scale. In *AKBC-WEKEX*, pages 84–88.

Raymond J. Mooney. 2007. Learning for semantic parsing. In *CICLing*, pages 311–324.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *ACL*, pages 160–167.

Hoifung Poon. 2013. Grounded unsupervised semantic parsing. In *ACL*, pages 933–943.

Yuk Wah Wong and Raymond J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *HLT-NAACL*.

Yuk Wah Wong and Raymond J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *ACL*.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI, Vol. 2*, pages 1050–1055.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*, pages 658–666.

Luke S. Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *EMNLP-CoNLL*, pages 678–687.

Discourse Complements Lexical Semantics for Non-factoid Answer Reranking

Peter Jansen and **Mihai Surdeanu**
University of Arizona
Tucson, AZ, USA
{pajansen,msurdeanu}
@email.arizona.edu

Peter Clark
Allen Institute for Artificial Intelligence
Seattle, WA, USA
peterc@allenai.org

Abstract

We propose a robust answer reranking model for non-factoid questions that integrates lexical semantics with discourse information, driven by two representations of discourse: a shallow representation centered around discourse markers, and a deep one based on Rhetorical Structure Theory. We evaluate the proposed model on two corpora from different genres and domains: one from Yahoo! Answers and one from the biology domain, and two types of non-factoid questions: manner and reason. We experimentally demonstrate that the discourse structure of non-factoid answers provides information that is complementary to lexical semantic similarity between question and answer, improving performance up to 24% (relative) over a state-of-the-art model that exploits lexical semantic similarity alone. We further demonstrate excellent domain transfer of discourse information, suggesting these discourse features have general utility to non-factoid question answering.

1 Introduction

Driven by several international evaluations and workshops such as the Text REtrieval Conference (TREC)[1] and the Cross Language Evaluation Forum (CLEF),[2] the task of question answering (QA) has received considerable attention. However, most of this effort has focused on factoid questions rather than more complex non-factoid (NF) questions, such as manner, reason, or causation questions. Moreover, the vast majority of QA models explore only local linguistic structures, such as syntactic dependencies or semantic role frames, which are generally restricted to individual sentences. This is problematic for NF QA, where questions are answered not by atomic facts, but by larger cross-sentence conceptual structures that convey the desired answers. Thus, to answer NF questions, one needs a model of what these answer structures look like.

Driven by this observation, our main hypothesis is that the discourse structure of NF answers provides complementary information to state-of-the-art QA models that measure the similarity (either lexical and/or semantic) between question and answer. We propose a novel answer reranking (AR) model that combines lexical semantics (LS) with discourse information, driven by two representations of discourse: a shallow representation centered around discourse markers and surface text information, and a deep one based on the Rhetorical Structure Theory (RST) discourse framework (Mann and Thompson, 1988). To the best of our knowledge, this work is the first to systematically explore within- and cross-sentence structured discourse features for NF AR. The contributions of this work are:

1. We demonstrate that modeling discourse is greatly beneficial for NF AR for two types of NF questions, manner (*"how"*) and reason (*"why"*), across two large datasets from different genres and domains – one from the community question-answering (CQA) site of Yahoo! Answers[3], and one from a biology textbook. Our results show statistically significant improvements of up to 24% on top of state-of-the-art LS models (Yih et al., 2013).
2. We demonstrate that both shallow and deep discourse representations are useful, and, in general, their combination performs best.
3. We show that discourse-based QA models using inter-sentence features considerably out-

[1] http://trec.nist.gov
[2] http://www.clef-initiative.eu
[3] http://answers.yahoo.com

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 977–986,
Baltimore, Maryland, USA, June 23-25 2014.

perform single-sentence models when answers span multiple sentences.

4. We demonstrate good domain transfer performance between these corpora, suggesting that answer discourse structures are largely independent of domain, and thus broadly applicable to NF QA.

2 Related Work

The body of work on factoid QA is too broad to be discussed here (see, e.g., the TREC workshops for an overview). However, in the context of LS, Yih et al. (2013) recently addressed the problem of answer sentence selection and demonstrated that LS models, including recurrent neural network language models (RNNLM), have a higher contribution to overall performance than exploiting syntactic analysis. We extend this work by showing that discourse models coupled with LS achieve the best performance for NF AR.

The related work on NF QA is considerably more scarce, but several trends are clear. First, most NF QA approaches tend to use multiple similarity models (information retrieval or alignment) as features in discriminative rerankers (Riezler et al., 2007; Higashinaka and Isozaki, 2008; Verberne et al., 2010; Surdeanu et al., 2011). Second, and more relevant to this work, all these approaches focus either on bag-of-word representations or linguistic structures that are restricted to single sentences (e.g., syntactic dependencies, semantic roles, or standalone discourse cue phrases).

Answering *how* questions using a single discourse marker, *by*, was previously explored by Prager et al. (2000), who searched for *by* followed by a present participle (e.g. *by *ing*) to elevate answer candidates in a ranking framework. Verberne et al. (2011) extracted 47 cue phrases such as *because* from a small collection of web documents, and used the cosine similarity between an answer candidate and a bag of words containing these cue phrases as a single feature in their reranking model for non-factoid *why* QA. Extending this, Oh et al. (2013) built a classifier to identify causal relations using a small set of cue phrases (e.g., *because* and *is caused by*). This classifier was then used to extract instances of causal relations in answer candidates, which were turned into features in a reranking model for Japanese *why* QA.

In terms of discourse parsing, Verberne et al. (2007) conducted an initial evaluation of the utility of RST structures to *why* QA by evaluating

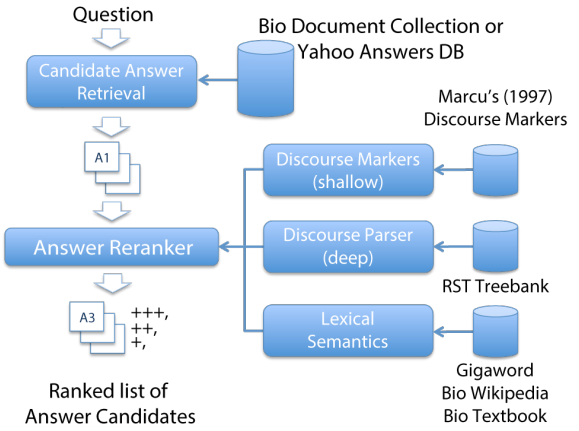


Figure 1: Architecture of the reranking framework for QA.

performance on a small sample of seven WSJ articles drawn from the RST Treebank (Carlson et al., 2003). They later concluded that while discourse parsing appears to be useful for QA, automated discourse parsing tools are required before this approach can be tested at scale (Verberne et al., 2010). Inspired by this previous work and recent work in discourse parsing (Feng and Hirst, 2012), our work is the first to systematically explore structured discourse features driven by several discourse representations, combine discourse with lexical semantic models, and evaluate these representations on thousands of questions using both in-domain and cross-domain experiments.

3 Approach

The proposed answer reranking component is embedded in the QA framework illustrated in Figure 1. This framework functions in two distinct scenarios, which use the same AR model, but differ in the way candidate answers are retrieved:

CQA: In this scenario, the task is defined as reranking all the user-posted answers for a particular question to boost the community-selected best answer to the top position. This is a commonly used setup in the CQA community (Wang et al., 2009).[4] Thus, for a given question, all its answers are fetched from the answer collection, and an initial ranking is constructed based on the cosine similarity between theirs and the question's lemma vector representations, with lemmas weighted using *tf.idf* (Ch. 6, (Manning et al., 2008)).

[4] Although most of these works use shallow textual features and focus mostly on meta data, e.g., number of votes for a particular answer. Here we use no meta data and rely solely on linguistic features.

Traditional QA: In this scenario answers are dynamically constructed from larger documents (Pasca, 2001). We use this setup to answer questions from a biology textbook, where each section is indexed as a standalone document, and each paragraph in a given document is considered as a candidate answer. We implemented the document indexing and retrieval stage using Lucene[5]. The candidate answers are scored using a linear interpolation of two cosine similarity scores: one between the entire parent document and question (to model global context), and a second between the answer candidate and question (for local context).[6] Because the number of answer candidates is typically large (e.g., equal to the number of paragraphs in the textbook), we return the N top candidates with the highest scores.

These answer candidates are then passed to the answer reranking component, the focus of this work. AR analyzes the candidates using more expensive techniques to extract discourse and LS features (detailed in §4), and these features are then used in concert with a learning framework to rerank the candidates and elevate correct answers to higher positions. For the learning framework, we used SVM^{rank}, a variant of Support Vector Machines for structured output adapted to ranking problems.[7] In addition to these features, each reranker also includes a single feature containing the score of each candidate, as computed by the above candidate retrieval (CR) component.[8]

4 Models and Features

We propose two separate discourse representation schemes – one shallow, centered around discourse markers, and one deep, based on RST.

4.1 Discourse Marker Model

The discourse marker model (DMM) extracts cross-sentence discourse structures centered around a discourse marker. This extraction process is illustrated in the top part of Figure 2. These structures are represented using three components: (1) A **discourse marker** from Daniel Marcu's list (see Appendix B in Marcu (1997)), that serves as a divisive boundary between sentences. Examples of these markers include *and, in, that, for, if, as, not, by,* and *but*; (2) **two marker arguments**, i.e., text segments before and after the marker, labeled to indicate if they are related to the question text or not; and (3) **a sentence range** around the marker, which defines the length of these segments (e.g., ± 2 sentences). For example, a marker feature may take the form of: `QSEG BY OTHER SR2`, which means that the the marker *by* has been detected in an answer candidate. Further, the text preceeding *by* matches text from the question (and is therefore labeled `QSEG`), while the text after *by* differs considerably from the question text, and is labeled `OTHER`. In this particular example, the scope of this similarity matching occurs over a span of ± 2 sentences around the marker.

Note that our marker arguments are akin to EDUs in RST, but, in this shallow representation, they are simply constructed around discourse markers and bound by an arbitrary sentence range.

Argument Labels: We label marker arguments based on their similarity to question content. If text before or after a marker out to a given sentence range matches the entire text of the question (with a cosine similarity score larger than a threshold), that argument takes on the label `QSEG`, or `OTHER` otherwise. In this way the features are only partially lexicalized with the discourse markers. Argument labels indicate only if lemmas from the question were found in a discourse structure present in an answer candidate, and do not speak to the specific lemmas that were found. We show in §5 that these lightly lexicalized features perform well in domain and transfer between domains. We explore other argument labeling strategies in §5.7.

Feature Values: Our reranking framework uses real-valued features. The values of the discourse features are the mean of the similarity scores (e.g., cosine similarity using *tf.idf* weighting) of the two marker arguments and the corresponding question. For example, the value of the `QSEG BY QSEG SR1` feature in Figure 2 is the average of the cosine similarities of the question text with the answer texts before/after *by* out to a distance of one sentence before/after the marker.

It is important to note that these discourse features are more expressive than features based on discourse markers alone (Higashinaka and Isozaki, 2008; Verberne et al., 2010). First,

[5] `http://lucene.apache.org`

[6] We empirically observed that this combination of scores performs better than using solely the cosine similarity between the answer and question.

[7] `http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html`

[8] Including these scores as features in the reranker model is a common strategy that ensures that the reranker takes advantage of the analysis already performed by the CR model.

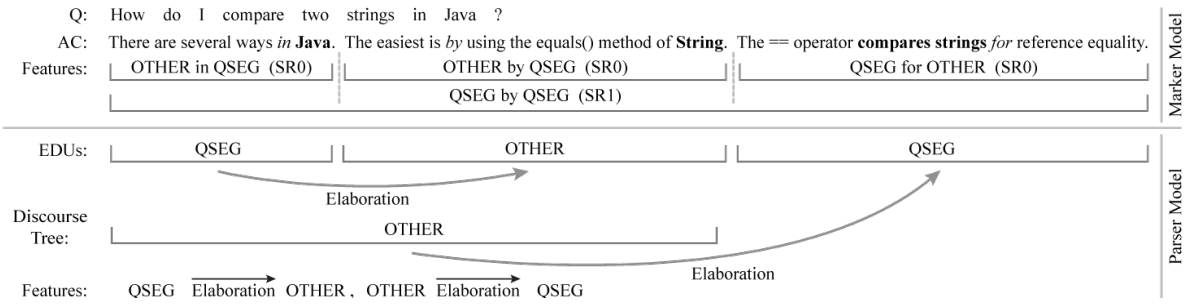


Figure 2: **Top:** Example feature generation for the discourse marker model, for one question (Q) and one answer candidate (AC). Answer candidates are searched for discourse markers (*italic*) and question word matches (**bold**), which are used to generate features both within-sentence (SR0), and ± 1 sentence (SR1). The actual DMM exhaustively generates features for all markers and all sentence ranges. Here we show just a few for brevity. **Bottom:** Example feature generation for the discourse parser model using the output of an actual discourse parser. The DPM creates one feature for each individual discourse relation.

the argument sequences used here capture cross-sentence discourse structures. Second, these features model the intensity of the match between the text surrounding the discourse structure and the question text using both the assigned argument labels and the feature values.

4.2 Discourse Parser Model

The discourse parser model (DPM) is based on the RST discourse framework (Mann and Thompson, 1988). In RST, the text is segmented into a sequence of non-overlapping fragments called elementary discourse units (EDUs), and binary discourse relations recursively connect neighboring units. Most relations are *hypotactic*, where one of the units in the relation (the *nucleus*) is considered more important than the other (the *satellite*). A few relations are *paratactic*, where both participants have equal importance. In the bottom part of Figure 2, we show hypotactic relations as directed arrows, from the nucleus to the satellite. In this work, we construct the RST discourse trees using the parser of Feng and Hirst (2012).

Relying on a proper discourse framework facilitates the modeling of the numerous implicit relations that are not driven by discourse markers (see Ch. 21 in Jurafsky and Martin (2009)). However, this also introduces noise because discourse analysis is a complex task and discourse parsers are not perfect. To mitigate this, we used a simple feature generation strategy, which creates one feature for each individual discourse relation by concatenating the relation type with the labels of the discourse units participating in it. To this end, for every relation, we extract the entire text dominated by each of its arguments, and we generate labels for the two participants in the relation using the same strategy as the DMM (based on the similarity with the question content). Similar to the DMM, these features take real values obtained by averaging the cosine similarity of the arguments with the question content.[9] Fig. 2 shows several such features, created around two RST `Elaboration` relations, indicating that the latter sentences expand on the information at the beginning of the answer. Other common relations include `Attribution`, `Contrast`, `Background`, and `Evaluation`.

4.3 Lexical Semantics Model

Inspired by the work of Yih et al. (2013), we include lexical semantics in our reranking model. Several of their proposed models rely on proprietary data; here we focus on LS models that rely on open-source data and frameworks. In particular, we use the recurrent neural network language model (RNNLM) of Mikolov et al. (2013; 2010). Like any language model, a RNNLM estimates the probability of observing a word given the preceding context, but, in this process, it learns word embeddings into a latent, conceptual space with a fixed number of dimensions. Consequently, related words tend to have vectors that are close to each other in this space.

We derive two LS measures from these vectors, which are then are included as features in the reranker. The first is a measure of the overall LS similarity of the question and answer can-

[9] We investigated more complex features, e.g., by exploring depths of two and three in the discourse tree, and also models that relied on tree kernels over these trees, but none improved upon this simple representation. This suggests that, in the domains explored here, there is a degree of noise introduced by the discourse parser, and the simple features proposed here are the best strategy to avoid overfitting on it.

didate, which is computed as the cosine similarity between the two composite vectors of the question and the answer candidate. These composite vectors are assembled by summing the vectors for individual question (or answer candidate) words, and re-normalizing this composite vector to unit length. Both this overall similarity score, as well as the average pairwise cosine similarity between each word in the question and answer candidate, serve as features.

5 Experiments

5.1 Data

To test the utility of our approach, we experimented with the two QA scenarios introduced in §3 using the following two datasets:

Yahoo! Answers Corpus (YA): Yahoo! Answers[10] is an open domain community-generated QA site, with questions and answers that span formal and precise to informal and ambiguous language. Due to the speed limitations of the discourse parser, we randomly drew 10,000 QA pairs from the corpus of *how* questions described by Surdeanu et al. (2011) using their filtering criteria, with the additional criterion that answers had to contain at least four community-generated answers, one of which was voted as the top answer. The number of answers to each question ranged from 4 to over 50, with the average 9.[11]

Biology Textbook Corpus (Bio): This corpus focuses on the domain of cellular biology, and consists of 185 *how* and 193 *why* questions handcrafted by a domain expert. Each question has one or more gold answers identified in Campbell's Biology (Reece et al., 2011), a popular undergraduate text. The entire biology text (at paragraph granularity) serves as the possible set of answers. Note that while our system retrieves answers at paragraph granularity, the expert was not constrained in any way during the annotation process, so gold answers might be smaller than a paragraph or span multiple paragraphs. This complicates evaluation metrics on this dataset (see §5.3).

[10] `http://answers.yahoo.com`

[11] Note that our experimental setup, i.e., reranking all the answers provided for each question, is different from that of Surdeanu et al. For each question, they retrieved candidate answers from all answers voted as best for some question in the collection. The setup in this paper, commonly used in the CQA community (Wang et al., 2009), is more relevant here because it includes both high and low quality answers.

For the YA CQA corpora, 50% of QA pairs were used for training, 25% for development, and 25% for test. Because of the small size of the Bio corpus, it was evaluated using 5-fold cross-validation, with three folds for training, one for development, and one for test.

The following additional resources were used:

Discourse Markers: A set of 75 high-frequency[12] single-word discourse markers were extracted from Marcu's (1997) list of cue phrases, and used for feature generation in DMM. These discourse markers are extremely common in the answer corpora – for example, the YA corpus contains an average of 7 markers per answer.

Discourse Trees: We generated all discourse trees using the parser of Feng and Hirst (2012). For YA, we parsed entire answers. For Bio, we parsed individual paragraphs. Note that, because these domains are considerably different from the RST Treebank, the parser fails to produce a tree on a large number of answer candidates: 6.2% for YA, and 41.1% for Bio. In these situations, we constructed artificial discourse trees using a right-attachment heuristic and a single relation label `X`.

Lexical Semantics: We trained two different RNNLMs for this work. First, for the YA experiments we trained an open-domain RNNLM using the entire Gigaword corpus of approximately 4G words.[13] For the Bio experiments, we trained a domain specific RNNLM over a concatenation of the textbook and a subset of Wikipedia specific to biology. The latter was created by extracting: (a) pages matching a word/phrase in a glossary of biology (derived from the textbook); plus (b) pages hyperlinked from (a) that are also tagged as being in a small set of (hand-selected) biology-related categories. The combined dataset contains 7.7M words. For all RNNLMs we used 200-dimensional vectors.

5.2 Hyper Parameter Tuning

The following hyper parameters were tuned using grid search to maximize P@1 on each development partition: (a) the segment matching thresholds that determine the minimum cosine similarity between an answer segment and a question for the segment to be labeled `QSEG`; and (b)

[12] We selected all cue phrases with more than 100 occurrences in the Brown corpus.

[13] LDC catalog number LDC2012T21

#	Model/Features	P@1	P@1 Impr.	MRR	MRR Impr.
	YA Corpus				
1	Random Baseline	14.29		26.12	
2	CR Baseline	19.57		43.14	
3	CR + DMM	24.05*	+23%	46.40*	+8%
4	CR + DPM	24.29*	+24%	46.81*	+9%
5	CR + DMM + DPM	**24.81***	**+27%**	**47.10***	**+9%**
6	CR + LS Baseline	26.57		49.31	
7	CR + LS + DMM	29.29*	+10%	50.99*	+3%
8	CR + LS + DPM	28.73*	+8%	50.77*	+3%
9	CR + LS + DMM + DPM	**30.49***	**+15%**	**51.89***	**+5%**
	Bio HOW				
10	CR Baseline	24.12		32.90	
11	CR + DMM	29.88*	+24%	38.88*	+18%
12	CR + DPM	28.93*	+20%	37.75*	+15%
13	CR + DMM + DPM	**30.43***	**+26%**	**39.28***	**+19%**
14	CR + LS Baseline	25.35		33.79	
15	CR + LS + DMM	30.09*	+19%	39.04*	+16%
16	CR + LS + DPM	28.50	+12%	37.58*	+11%
17	CR + LS + DMM + DPM	**30.68***	**+21%**	**39.44***	**+17%**
	Bio WHY				
18	CR Baseline	28.62		38.25	
19	CR + DMM	38.01*	+33%	46.39*	+21%
20	CR + DPM	38.62*	+35%	46.85*	+23%
21	CR + DMM + DPM	**39.36***	**+38%**	**47.64***	**+25%**
22	CR + LS Baseline	31.73		39.89	
23	CR + LS + DMM	38.60*	+22%	46.41*	+16%
24	CR + LS + DPM	**39.45***	**+24%**	47.38*	+19%
25	CR + LS + DMM + DPM	39.32*	+24%	**47.86***	**+20%**

Table 1: Overall results across three datasets. The improvements in each section are computed relative to their respective baseline (CR or CR + LS). Bold font indicates the best score in a given column. * indicates that a score is significantly better ($p < 0.05$) than the score of the corresponding baseline. All significance tests were implemented using one-tailed non-parametric bootstrap resampling using 10,000 iterations.

SVM^{rank}'s regularization parameter C. For all experiments, the sentence range parameter (`SRx`) for DMM ranged from 0 (within sentence) to ± 3 sentences.[14]

5.3 Evaluation Metrics

For YA, we used the standard implementations for P@1 and mean reciprocal rank (MRR) (Manning et al., 2008). In the Bio corpus, because answer candidates are not guaranteed to match gold annotations exactly, these metrics do not immediately apply. We adapted them to this dataset by weighing each answer by its overlap with gold answers, where overlap is measured as the highest F1 score between the candidate and a gold answer. Thus, P@1 reduces to this F1 score for the top answer. For MRR, we used the rank of the candidate with the highest overlap score, weighed by the inverse of the rank. For example, if the best answer for a question appears at rank 2 with an F1 score of 0.3, the corresponding MRR score is $0.3/2$.

[14]This was only limited to reduce the combinatorial expansion of feature generation, and in principle could be set much broader.

5.4 Overall Results

Table 1 analyzes the performance of the proposed reranking model on the three datasets and against two baselines. The first baseline sorts the candidate answers in descending order of the scores produced by the candidate retrieval (CR) module. The second baseline (CR + LS) trains a reranking model without discourse, using just the CR and LS features. For YA, we include an additional baseline that selects an answer randomly. We list multiple versions of the proposed reranking model, broken down by the features used. For Bio, we retrieved the top 20 answer candidates in CR. At this setting, the oracle performance (i.e., the performance with perfect reranking of the 20 candidates) was 69.6% P@1 for Bio HOW, and 72.3% P@1 for Bio WHY. These relatively low oracle scores, which serve as a performance ceiling for our approach, highlight the difficulty of the task. For YA, we used all answers provided for each given question. For all experiments we used a linear SVM kernel.[15]

Examining Table 1, several trends are clear. Both discourse models significantly increase both P@1 and MRR performance over all baselines broadly across genre, domain, and question types. More specifically, DMM and DPM show similar performance benefits when used individually, but their combination generally outperforms the individual models, illustrating the fact that the two models capture related but different discourse information. This is a motivating result for discourse analysis, especially considering that the discourse parser was trained on a domain different from the corpora used here.

Lexical semantic features increase performance for all settings, but demonstrate far more utility to the open-domain YA corpus. This disparity is likely due to the difficulty in assembling LS training data at an appropriate level for the biology corpus, contrasted with the relative abundance of large scale open-domain lexical semantic resources. For the YA corpus, where lexical semantics showed the most benefit, simply adding

[15]The performance of all models can ultimately be increased by using more sophisticated learning frameworks, and considering more answer candidates in CR (for Bio). For example, SVMs with polynomial kernels of degree two showed approximately half a percent (absolute) performance gain over the linear kernel. However, this came at the expense of an experiment runtime about an order of magnitude larger. Experiments with more answer candidates in Bio showed similar trends to the results reported.

Q	How does myelination affect action potentials?
$A_{baseline}$	The major selective advantage of myelination is its space efficiency. A myelinated axon 20 microns in diameter has a conduction speed faster than that of a squid giant axon [...]. Furthermore, more than 2,000 of those myelinated axons can be packed into the space occupied by just one giant axon.
A_{rerank}	A nerve impulse travels [...] to the synaptic terminals by propagation of a series action potentials along the axon. The speed of conduction increases [...] with myelination. Action potentials in myelinated axons jump between the nodes of Ranvier, a process called saltatory conduction.

Table 2: An example question from the Biology corpus where the correct answer is elevated to the top position by the discourse model. $A_{baseline}$ is the top answer proposed by the CR + LS baseline, which is incorrect, whereas A_{rerank} is the correct answer boosted to the top after reranking. [...] indicates non-essential text that was removed for space.

Range	Bio HOW	Bio WHY	YA
CR + LS + DMM + DPM			
within-sentence	+0.8%	+8.4%	+13.1%
full model	+21.0%*	+23.9%*	+14.8%

Table 3: Relative P@1 performance increase over the CR + LS baseline for a model containing only intra-sentence features, compared to the full model.

LS features to the CR baseline increases baseline P@1 performance from 19.57 to 26.57, a +36% relative improvement. Most importantly, comparing lines 5 and 9 with their respective baselines (lines 2 and 6, respectively) indicates that LS is largely orthogonal to discourse. Line 5, the top-performing model with discourse but without LS outperforms the CR baseline by +5.24 absolute P@1 improvement. Similarly, line 9, the top-performing model that combines discourse with LS has a +5.69 absolute P@1 improvement over the CR + LS baseline. That this absolute performance increase is nearly identical indicates that LS features are complementary to and additive with the full discourse model. Indeed, an analysis of the questions improved by discourse vs. LS (line 5 vs. 6) showed that the intersection of the two sets is low (approximately a third of each set).

Finally, while the discourse models perform well for HOW or *manner* questions, performance on Bio WHY corpus suggests that *reason* questions are particularly amenable to discourse analysis. Relative improvements on WHY questions reach +38% (without LS) and +24% (with LS), with absolute performance on these non-factoid questions jumping from 28% to nearly 40% P@1.

Table 2 shows one example where discourse helps boost the correct answer to the top position. In this example, the correct answer contains multiple `Elaboration` relations that are both cross sentence (e.g., between the first two sentences) and intra-sentence (e.g., between the first part of the second sentence and the phrase "with myelination"). Model features associated with `Elaboration` relations are ranked highly by the learned model. In contrast, the answer preferred by the baseline contains mostly `Joint` relations, which "represent the lack of a rhetorical relation between the two nuclei" (Mann and Thompson, 1988) and have very small weights in the model.

5.5 Intra vs. Inter-sentence Features

To tease apart the relative contribution of discourse features that occur only within a single sentence versus features that span multiple sentences, we examined the performance of the full model when using only intra-sentence features, i.e., `SR0` features for DMM, and features based on discourse relations where both EDUs appear in the same sentence for DPM, versus the full intersentence models. The results are shown in Table 3.

For the Bio corpus where answer candidates consist of entire paragraphs of a biology text, overall performance is dominated by inter-sentence discourse features. Conversely, for YA, a large proportion of performance comes from features that span only a single sentence. This is caused by the fact that YA answers are far shorter and of variable grammatical quality, with 39% of answer candidates consisting of only a single sentence, and 57% containing two or fewer sentences. All in all, this experiment emphasizes that modeling both intra- and inter-sentence discourse (where available) is beneficial for non-factoid QA.

5.6 Domain Transfer

Because these discourse models appear to capture high-level information about answer structures, we hypothesize that the models should make use of many of the same discourse features, even when training on data from different domains. Table 4 shows that of the highest-weighted SVM features learned when training models for HOW questions on YA and Bio, many are shared (e.g., 56.5% of the features in the top half of both DPMs are shared), suggesting that a core set of discourse features may be of utility across domains.

To test the generality of these features, we performed a transfer study where the full model was trained and tuned on the open-domain YA corpus, then evaluated as is on Bio HOW. This is

Model	Top 10%	Top 25%	Top 50%
DMM	20.2%	33.2%	49.4%
DPM	22.2%	39.1%	56.5%

Table 4: Percentage of top features with the highest SVM weights that are shared between Bio HOW and YA models.

a somewhat radical setup, where the target corpus has both a different genre (formal text vs. CQA) and different domain (biology vs. open domain). These experiments were performed in several groups: both with and without LS features, as well as using either a single SVM or an ensemble model that linearly interpolates the predictions of two SVM classifiers (one each for DMM and DPM).[16] The results are summarized in Table 5.

The transferred models always outperform the baselines, but only the ensemble model's improvement is statistically significant. This confirms existing evidence that ensemble models perform better cross-domain because they overfit less (Domingos, 2012; Hastie et al., 2009). The ensemble model without LS (third line) has a nearly identical P@1 score as the equivalent in-domain model (line 13 in Table 1), while slightly surpassing in-domain MRR performance. To the best of our knowledge, this is one of the most striking demonstrations of domain transfer in answer ranking for non-factoid QA, and highlights the generality of these discourse features in identifying answer structures across domains and genres.

The results of the transferred models that include LS features are slightly lower, but still approach statistical significance for P@1 and are significant for MRR. We hypothesize that the limited transfer observed for models with LS compared to their counterparts without LS is due to the disparity in the size and utility of the biology LS training data compared to the open-domain LS resources. The open-domain YA model learns to place more weight on LS features, which are unable to provide the same utility in the biology domain.

5.7 Integrating Discourse and LS

So far, we have treated LS and discourse as distinct features in the reranking model, However, given that LS features greatly improve the CR baseline, we hypothesize that a natural extension

[16]The interpolation parameter was tuned on the YA development corpus. The in-domain performance of the ensemble model is similar to that of the single classifier in both YA and Bio HOW so we omit these results here for simplicity.

Model/Features	P@1	P@1 Impr.	MRR	MRR Impr.
Transfer: YA → Bio HOW				
CR Baseline	24.12		32.90	
CR + DMM + DPM	27.13	+13%	36.36†	+11%
(CR + DMM) ∪ (CR + DPM)	**30.10***	**+25%**	**39.62***	**+20%**
CR + LS Baseline	25.35		33.79	
CR + LS + DMM + DPM	25.79	+2%	35.58	+5%
(CR + LS + DMM) ∪ (CR + LS + DPM)	**29.54**†	**+17%**	**38.68***	**+15%**

Table 5: Transfer performance from YA to Bio HOW for single classifiers and ensembles (denoted with a ∪). † indicates approaching statistical significance with $p = 0.07$ or 0.06.

to the discourse models would be to make use of LS similarity (in addition to the traditional information retrieval similarity) to label discourse segments. For example, for the question *"How do cells replicate?"*, answer discourse segments containing LS associates of *cell* and *replicate*, e.g., *nucleus, membrane, genetic*, and *duplicate*, should be considered as related to the question (i.e., be labeled QSEG). We implemented two such models, denoted DMM$_{LS}$ and DPM$_{LS}$, by replacing the component that assigns argument labels with one that relies on LS. Specifically, as in §4.3, we compute the cosine similarity between the composite LS vectors of the question text and each marker argument (in DMM) or EDU (in DPM), and label the corresponding answer segment QSEG if this score is higher than a threshold, or OTHER otherwise. This way, the DMM and DPM features jointly capture discourse structures and semantic similarity between answer segments and question.

To test this, we use the YA corpus, which has the best-performing LS model. Because we are adding two new discourse models, we now tune four segment matching thresholds, one for each of the DMM, DPM, DMM$_{LS}$, and DPM$_{LS}$ models.[17] The results are shown in Table 6. These results demonstrate that incorporating LS in the discourse models further increases performance for all configurations, nearly doubling the relative performance benefits over models that do not integrate LS and discourse (compare with lines 6–9 of Table 1). For example, the last model in the table, which combines four discourse representations, improves P@1 by 24%, whereas the equivalent model without this integration (line 9 in Table 1) outperforms the baseline by only 15%.

[17]These hyperparameters were tuned on the development corpus, and were found to be stable over broad ranges.

Model Features	P@1	P@1 Impr.	MRR	MRR Impr.
CR + LS Baseline	26.57		49.31	
CR + LS + DMM + DMM_{LS}	32.41*	+22%	53.55*	+9%
CR + LS + DPM + DPM_{LS}	31.21*	+18%	52.50*	+7%
CR + LS + DMM + DPM + DMM_{LS} + DPM_{LS}	**32.93***	**+24%**	**53.91***	**+9%**

Table 6: YA results with integrated discourse and LS.

5.8 Error Analysis

We performed an error analysis of the full QA model (CR + LS + DMM + DPM) across the entire Bio corpus (lines 17 and 25 from Table 1). We chose the Bio setup for this analysis because it is more complex than the CQA one: here gold answers may have a granularity completely different from what the system choses as best answers (in our particular case, the QA system is currently limited to answers consisting of single paragraphs, whereas gold answers may be of any size).

Here, 94 of the 378 Bio HOW and WHY questions have improved answer scores, while 36 have reduced performance relative to the CR baseline. Of these 36 questions where answer scores decreased, nearly two thirds were directly related to the paragraph granularity of the candidate answer retrieval (see §5.1):

Same Subsection (50%): In these cases, the model selected an on-topic answer paragraph in the same subsection of the textbook as a gold answer. Often times this paragraph directly preceded or followed the gold answer.

Answer Window Size (14%): Here, both the CR and full model chose a paragraph containing a different gold answer. However, as discussed, gold answers may unevenly straddle paragraph boundaries, and the paragraph chosen by the model happened to have a somewhat lower overlap with its gold answer than the one chosen by the baseline.

Similar Topic (25%): The model chose a paragraph that had a similar topic to the question, but doesn't answer the question. These are challenging errors, often associated with short questions (e.g. *"How does HIV work?"*) that provide few keywords. In these cases, discourse features tend to dominate, and shift the focus towards answers that have many discourse structures deemed relevant. For example, for the above question, the model chose a paragraph containing many discourse structures positively correlated with high-quality answers, but which describes the origins of HIV instead of how the virus enters a cell.

Similar Words, Different Topic (8%): The model chose a paragraph that had many of the same words as the question, but is on a different topic. For example, for the question *"How are fossil fuels formed, and why do they contain so much energy?"*, the model selected an answer that mentions fossil fuels in a larger discussion of human ecological footprints. Here, the matching of both keywords and discourse structures shifted the answer towards a different, incorrect topic.

Finally, in one case (3%), the model identified an answer paragraph that contained a gold answer, but was missed by the domain expert annotator.

In summary, this analysis suggests that, for the majority of errors, the QA system selects an answer that is both topical and adjacent to a gold answer selected by the domain expert. This suggests that most errors are minor and are driven by current limitations of our answer boundary selection mechanism, rather than the inherent limitations of the discourse model.

6 Conclusions

This work focuses on two important aspects of answer reranking for non-factoid QA: similarity between question and answer content, and answer structure. While the former has been addressed with a variety of lexical-semantic models, the latter has received little attention. Here we show how to model answer structures using discourse and how to integrate the two aspects into a holistic framework. Empirically we show that modeling answer discourse structures is complementary to modeling lexical semantic similarity and that the best performance is obtained when they are tightly integrated. We evaluate the proposed approach on multiple genres and question types and obtain benefits of up to 24% relative improvement over a strong baseline that combines information retrieval and lexical semantics. We further demonstrate that answer discourse structures are largely independent of domain and transfer well, even between radically different datasets.

This work is open source and available at: `http://nlp.sista.arizona.edu/releases/acl2014`.

Acknowledgements

We thank the Allen Institute for Artificial Intelligence for funding this work. We would also like to thank the three anonymous reviewers for their helpful comments and suggestions.

References

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2003. Building a Discourse-Tagged Corpus in the Framework of Rhetorical Structure Theory. In Jan van Kuppevelt and Ronnie Smith, editors, *Current Directions in Discourse and Dialogue*, pages 85–112. Kluwer Academic Publishers.

Pedro Domingos. 2012. A few useful things to know about machine learning. *Communications of the ACM*, 55(10).

Vanessa Wei Feng and Graeme Hirst. 2012. Text-level discourse parsing with rich linguistic features. In *Proceedings of the Association for Computational Linguistics*.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer.

Ryuichiro Higashinaka and Hideki Isozaki. 2008. Corpus-based question answering for why-questions. In *Proceedings of the Proceedings of the Third International Joint Conference on Natural Language Processing (IJCNLP)*, pages 418–425, Hyderabad, India.

Dan Jurafsky and James H. Martin. 2009. *Speech and Language Processing, Second Edition*. Prentice Hall.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

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

Daniel Marcu. 1997. *The Rhetorical Parsing, Summarization, and Generation of Natural Language Texts*. Ph.D. thesis, University of Toronto.

Tomas Mikolov, Martin Karafiat, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Jong-Hoon Oh, Kentaro Torisawa, Chikara Hashimoto, Motoki Sano, Stijn De Saeger, and Kiyonori Ohtake. 2013. Why-question answering using intra- and inter-sentential causal relations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1733–1743, Sofia, Bulgaria, August. Association for Computational Linguistics.

Marius Pasca. 2001. *High-Performance, Open-Domain Question Answering from Large Text Collections*. Ph.D. thesis, Southern Methodist University.

John Prager, Eric Brown, Anni Coden, and Dragomir Radev. 2000. Question-answering by predictive annotation. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '00, pages 184–191, New York, NY, USA. ACM.

J.B. Reece, L.A. Urry, M.L. Cain, S.A. Wasserman, and P.V. Minorsky. 2011. *Campbell Biology*. Pearson Benjamin Cummings.

Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal, and Yi Liu. 2007. Statistical machine translation for query expansion in answer retrieval. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 464–471, Prague, Czech Republic.

Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. 2011. Learning to rank answers to non-factoid questions from web collections. *Computational Linguistics*, 37(2):351–383.

Susan Verberne, Lou Boves, Nelleke Oostdijk, Peter-Arno Coppen, et al. 2007. Discourse-based answering of why-questions. *Traitement Automatique des Langues, Discours et document: traitements automatiques*, 47(2):21–41.

Suzan Verberne, Lou Boves, Nelleke Oostdijk, and Peter-Arno Coppen. 2010. What is not in the bag of words for why-qa? *Computational Linguistics*, 36(2):229–245.

Suzan Verberne, Hans Halteren, Daphne Theijssen, Stephan Raaijmakers, and Lou Boves. 2011. Learning to rank for why-question answering. *Inf. Retr.*, 14(2):107–132, April.

Xin-Jing Wang, Xudong Tu, Dan Feng, and Lei Zhang. 2009. Ranking community answers by modeling question-answer relationships via analogical reasoning. In *Proceedings of the Annual ACM SIGIR Conference*.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*.

Toward Future Scenario Generation: Extracting Event Causality Exploiting Semantic Relation, Context, and Association Features

Chikara Hashimoto[*] **Kentaro Torisawa**[†] **Julien Kloetzer**[‡] **Motoki Sano**[§]
István Varga[¶] **Jong-Hoon Oh**[‖] **Yutaka Kidawara**[**]
[*†‡§‖**]National Institute of Information and Communications Technology, Kyoto, 619-0289, Japan
[¶]NEC Knowledge Discovery Research Laboratories, Nara, 630-0101, Japan
{[*]ch, [†]torisawa, [‡]julien, [§]msano, [‖]rovellia, [**]kidawara}@nict.go.jp

Abstract

We propose a supervised method of extracting event causalities like *conduct slash-and-burn agriculture→exacerbate desertification* from the web using semantic relation (between nouns), context, and association features. Experiments show that our method outperforms baselines that are based on state-of-the-art methods. We also propose methods of generating *future scenarios* like *conduct slash-and-burn agriculture→exacerbate desertification→increase Asian dust (from China)→asthma gets worse*. Experiments show that we can generate 50,000 scenarios with 68% precision. We also generated a scenario *deforestation continues→global warming worsens→sea temperatures rise→vibrio parahaemolyticus fouls (water)*, which is written in no document in our input web corpus crawled in 2007. But the vibrio risk due to global warming was observed in Baker-Austin et al. (2013). Thus, we "predicted" the future event sequence in a sense.

1 Introduction

The world can be seen as a network of causality where people, organizations, and other kinds of entities causally depend on each other. This network is so huge and complex that it is almost impossible for humans to exhaustively predict the consequences of a given event. Indeed, after the Great East Japan Earthquake in 2011, few expected that it would lead to an enormous trade deficit in Japan due to a sharp increase in energy imports. For effective decision making that carefully considers any form of future risks and chances, we need a system that helps humans do *scenario planning* (Schwartz, 1991), which is a decision-making scheme that examines possible future events and assesses their potential chances and risks. Our ultimate goal is to develop a system that supports scenario planning through generating possible future events using big data, which would contain what Donald Rumsfeld called "unknown unknowns"[1] (Torisawa et al., 2010).

To this end, we propose a supervised method of extracting such event causality as *conduct slash-and-burn agriculture→exacerbate desertification* and use its output to generate *future scenarios* (*scenarios*), which are chains of causality that have been or might be observed in this world like *conduct slash-and-burn agriculture→exacerbate desertification→increase Asian dust (from China)→asthma gets worse*. Note that, in this paper, *A→B* denotes that *A* causes *B*, which means that "*if A happens, the probability of B increases*." Our notion of causality should be interpreted probabilistically rather than logically.

Our method extracts event causality based on three assumptions that are embodied as features of our classifier. First, we assume that two nouns (e.g. *slash-and-burn agriculture* and *desertification*) that take some specific binary semantic relations (e.g. *A* CAUSES *B*) tend to constitute event causality if combined with two predicates (e.g. *conduct* and *exacerbate*). Note that semantic relations are not restricted to those directly relevant to causality like *A* CAUSES *B* but can be those that might seem irrelevant to causality like *A* IS AN INGREDIENT FOR *B* (e.g. *plutonium* and *atomic bomb* as in *plutonium is stolen→atomic bomb is made*). Our underlying intuition is the observation that event causality tends to hold between two entities linked by semantic relations which roughly entail that one entity strongly affects the other. Such semantic relations can be expressed by (otherwise unintuitive) patterns like *A* IS AN INGREDIENT FOR *B*. As such, semantic relations like the MATERIAL relation can also be useful. (See Sec-

[1]http://youtu.be/GiPe1OiKQuk

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 987–997,
Baltimore, Maryland, USA, June 23-25 2014.

tion 3.2.1 for a more intuitive explanation.)

Our second assumption is that there are grammatical contexts in which event causality is more likely to appear. We implement what we consider likely contexts for event causality as context features. For example, a likely context of event causality (underlined) would be: *CO2 levels rose, so climatic anomalies were observed*, while an unlikely context would be: *It remains uncertain whether if the recession is bottomed the declining birth rate is halted*. Useful context information includes the mood of the sentences (e.g., the uncertainty mood expressed by *uncertain* above), which is represented by lexical features (Section 3.2.2).

The last assumption embodied in our association features is that each word of the cause phrase must have a strong association (i.e., PMI, for example) with that of the effect phrase as *slash-and-burn agriculture* and *desertification* in the above example, as in Do et al. (2011).

Our method exploits these features on top of our base features such as nouns and predicates. Experiments using 600 million web pages (Akamine et al., 2010) show that our method outperforms baselines based on state-of-the-art methods (Do et al., 2011; Hashimoto et al., 2012) by more than 19% of average precision.

We require that event causality be *self-contained*, i.e., intelligible as causality without the sentences from which it was extracted. For example, *omit toothbrushing→get a cavity* is self-contained, but *omit toothbrushing→get a girlfriend* is not since this is not intelligible without a context: *He omitted toothbrushing every day and got a girlfriend who was a dental assistant of dental clinic he went to for his cavity*. This is important since future scenarios, which are generated by chaining event causality as described below, must be self-contained, unlike Hashimoto et al. (2012). To make event causality self-contained, we wrote guidelines for manually annotating training/development/test data. Annotators regarded as event causality only phrase pairs that were interpretable as event causality without contexts (i.e., self-contained). From the training data, our method seemed to successfully learn what self-contained event causality is.

Our scenario generation method generates scenarios by chaining extracted event causality; generating $A \rightarrow B \rightarrow C$ from $A \rightarrow B$ and $B \rightarrow C$. The challenge is that many acceptable scenarios are overlooked if we require the joint part of the chain (B above) to be an exact match. To increase the number of acceptable scenarios, our method identifies compatibility w.r.t causality between two phrases by a recently proposed semantic polarity, *excitation* (Hashimoto et al., 2012), which properly relaxes the chaining condition (Section 3.1 describes it). For example, our method can identify the compatibility between *sea temperatures are high* and *sea temperatures rise* to chain *global warming worsens→sea temperatures are high* and *sea temperatures rise→vibrio parahaemolyticus*[2] *fouls (water)*. Accordingly, we generated a scenario *deforestation continues→global warming worsens→sea temperatures rise→vibrio parahaemolyticus fouls (water)*, which is written in no document in our input web corpus that was crawled in 2007, but the vibrio risk due to global warming has actually been observed in the Baltic sea and reported in Baker-Austin et al. (2013). In a sense, we "predicted" the event sequence reported in 2013 by documents written in 2007. Our experiments also show that we generated 50,000 scenarios with 68% precision, which include *conduct terrorist operations→terrorist bombing occurs→cause fatalities and injuries→cause economic losses* and the above "*slash-and-burn agriculture*" scenario (Section 5.2). Neither is written in any document in our input corpus.

In this paper, our target language is Japanese. However, we believe that our ideas and methods are applicable to many languages. Examples are translated into English for ease of explanation. Supplementary notes of this paper are available at `http://khn.nict.go.jp/analysis/member/ch/acl2014-sup.pdf`.

2 Related Work

For **event causality extraction**, clues used by previous methods can roughly be categorized as lexico-syntactic patterns (Abe et al., 2008; Radinsky et al., 2012), words in context (Oh et al., 2013), associations among words (Torisawa, 2006; Riaz and Girju, 2010; Do et al., 2011), and predicate semantics (Hashimoto et al., 2012). Besides features similar to those described above, we propose semantic relation features[3] that include those that are not obviously related to causality. We show that such thorough exploitation of new and existing features leads to high performance.

[2] A bacterium in the sea causing food-poisoning.

[3] Radinsky et al. (2012) and Tanaka et al. (2012) used semantic relations to *generalize* acquired causality instances.

Other clues include shared arguments (Torisawa, 2006; Chambers and Jurafsky, 2008; Chambers and Jurafsky, 2009), which we ignore since we target event causality about two distinct entities.

To the best of our knowledge, **future scenario generation** is a new task, although previous works have addressed similar tasks (Radinsky et al., 2012; Radinsky and Horvitz, 2013). Neither involves chaining and restricts themselves to only one event causality step. Besides, the events they predict must be those for which similar events have previously been observed, and their method only applies to news domain.

Some of the scenarios we generated are written on no page in our input web corpus. Similarly, Tsuchida et al. (2011) generated semantic knowledge like causality that is written in no sentence. However, their method cannot combine more than two pieces of knowledge unlike ours, and their target knowledge consists of nouns, but ours consists of verb phrases, which are more informative.

Tanaka et al. (2013)'s web information analysis system provides a *what-happens-if QA* service, which is based on our scenario generation method.

3 Event Causality Extraction Method

This section describes our event causality extraction method. Section 3.1 describes how to extract event causality candidates, and Section 3.2 details our features. Section 3.3 shows how to rank event causality candidates.

3.1 Event Causality Candidate Extraction

We extract the event causality between two events represented by two phrases from single sentences that are dependency parsed.[4] We obtained sentences from 600 million web pages. Each phrase in the event causality must consist of a predicate with an argument position (*template*, hereafter) like *conduct X* and a noun like *slash-and-burn agriculture* that completes *X*. We also require the predicate of the cause phrase to syntactically depend on the effect phrase in the sentence from which the event causality was extracted; we guarantee this by verifying the dependencies of the original sentence. In Japanese, since the temporal order between events is usually determined by precedence in a sentence, we require the cause phrase to precede the effect phrase. For context feature extraction, the event causality candidates are accompanied by the original sentences from which they were extracted.

Excitation We only keep the event causality candidates each phrase of which consists of *excitation templates*, which have been shown to be effective for causality extraction (Hashimoto et al., 2012) and other semantic NLP tasks (Oh et al., 2013; Varga et al., 2013; Kloetzer et al., 2013a). Excitation is a semantic property of templates that classifies them into *excitatory*, *inhibitory*, and *neutral*. Excitatory templates such as *cause X* entail that the function, effect, purpose or role of their argument's referent is activated, enhanced, or manifested, while inhibitory templates such as *lower X* entail that it is deactivated or suppressed. Neutral ones like *proportional to X* belong to neither of them. We collectively call both excitatory and inhibitory templates excitation templates. We acquired 43,697 excitation templates by Hashimoto et al.'s method and the manual annotation of excitation template candidates.[5] We applied the excitation filter to all 272,025,401 event causality candidates from the web and 132,528,706 remained.

After applying additional filters (see Section A in the supplementary notes) including those based on a stop-word list and a causal connective list to remove unlikely event causality candidates that are not removed by the above filter, we finally acquired 2,451,254 event causality candidates.

3.2 Features for Event Causality Classifier

3.2.1 Semantic Relation Features

We hypothesize that two nouns with some particular semantic relations are more likely to constitute event causality. Below we describe the semantic relations that we believe are likely to constitute event causality.

CAUSATION is the causal relation between two entities and is expressed by binary patterns like *A* CAUSES *B*. *Deforestation* and *global warming* might complete the *A* and *B* slots. We manually collected 748 binary patterns for this relation. (See Section B in the supplementary notes for examples of our binary patterns.)

MATERIAL is the relation between a material and a product made of it (e.g. *plutonium* and

[4] We used a Japanese dependency parser called J.DepP (Yoshinaga and Kitsuregawa, 2009), available at `http://www.tkl.iis.u-tokyo.ac.jp/~ynaga/jdepp/`.

[5] Hashimoto et al.'s method constructs a network of templates based on their co-occurrence in web sentences with a small number of polarity-assigned seed templates and infers the polarity of all the templates in the network by a constraint solver based on the spin model (Takamura et al., 2005).

atomic bomb) and can be expressed by *A* IS MADE OF *B*. Its relation to event causality might seem unclear, but a material can be seen as a "cause" of a product. Indeed materials can participate in event causality with the help of such template pairs as *A is stolen→B is made* as in *plutonium is stolen→atomic bomb is made*. We manually collected 187 binary patterns for this relation.

NECESSITY's patterns include *A* IS NECESSARY FOR *B*, which can be filled with *verbal aptitude* and *ability to think*. Noun pairs with this relation can constitute event causality when combined with template pairs like *improve A→cultivate B*. We collected 257 patterns for this relation.

USE is the relation between means (or instruments) and the purpose for using them. *A* IS USED FOR *B* is a pattern of the relation, which can be filled with *e-mailer* and *exchanges of e-mail messages*. Note that means can be seen as "causing" or "realizing" the purpose of using the means in this relation, and actually event causality can be obtained by incorporating noun pairs of this relation into template pairs like *activate A→conduct B*. 2,178 patterns were collected for this relation.

PREVENTION is the relation expressed by patterns like *A* PREVENTS *B*, which can be filled with *toothbrushing* and *periodontal disease*. This relation is, so to speak, "negative CAUSATION" since the entity denoted by the noun completing the *A* slot makes the entity denoted by the *B* noun NOT realized. Such noun pairs mean event causality by substituting them into template pairs like *omit A→get B*. The number of patterns is 490.

The experiments in Section 5.1.1 show that not only CAUSATION and PREVENTION ("negative CAUSATION") but the other relations are also effective for event causality extraction.

In addition, we invented the EXCITATION relation that is expressed by binary patterns made of excitatory and inhibitory templates (Section 3.1). For instance, we make binary patterns *A* RISES *B* and *A* LOWERS *B* from excitatory template *rise X* and inhibitory template *lower X* respectively. The EXCITATION relation roughly means that *A* activates *B* (excitatory) or suppresses it (inhibitory). We simply add an additional argument position to each template in the 43,697 excitation templates to make binary patterns. We restricted the argument positions (represented by Japanese postpositions) of the *A* slot to either *ha* (topic marker), *ga* (nominative), or *de* (instrumental) and those of the *B* slot to either *ha*, *ga*, *de*, *wo* (accusative), or *ni* (dative), and obtained 55,881 patterns.

SR1: Binary pattern of our semantic relations that co-occurs with two nouns of an event causality candidate in our web corpus.

SR2: Semantic relation types (e.g CAUSATION and ENTAILMENT) of the binary pattern of SR1. EXCITATION is divided into six sub types based on the excitation polarity of the binary patterns, the argument positions, and the existence of causative markers. A CAUSATION pattern, *B* BY *A*, constitutes an independent relation called the BY relation.

Table 1: Semantic relation features.

Moreover, for broader coverage, we acquired binary patterns that entail or are entailed by one of the patterns of the above six semantic relations. Those patterns were acquired from our web corpus by Kloetzer et al. (2013b)'s method, which acquired 185 million entailment pairs with 80% precision from our web corpus and was used for contradiction acquisition (Kloetzer et al., 2013a). We acquired 335,837 patterns by this method. They are *class-dependent patterns*, which have semantic class restrictions on arguments. The semantic classes were obtained from our web corpus based on Kazama and Torisawa (2008). See De Saeger et al. (2009), De Saeger et al. (2011) and Kloetzer et al. (2013a) for more on our patterns. They collectively constitute the ENTAILMENT relation.

Table 1 shows our semantic relation features. To use them, we first make a database that records which noun pairs co-occur with each binary pattern. Then we check a noun pair (the nouns of the cause and effect phrases) for each event causality candidate, and give the candidate all the patterns in the database that co-occur with the noun pair.

3.2.2 Context Features

We believe that contexts exist where event causality candidates are more likely to appear, as described in Section 1. We developed features that capture the characteristics of likely contexts for Japanese event causality (See Section C in the supplementary notes). In a nutshell, they represent a connective (**C1** and **C2** in Section C), the distance between the elements of event causality candidate (**C3** and **C4**), words in context (**C5** to **C8**), the existence of adnominal modifier (**9** to **C10**), and the existence of additional arguments of cause and effect predicates (**C13** to **C20**), among others.

3.2.3 Association Features

These features measure the association strength between *slash-and-burn agriculture* and *deser-*

AC1: The CEA value, the sum of AC2, AC3, and AC4.
AC2: Do et al.'s S_{pp}. This is the association measure between predicates, which is the product of AC5, AC6 and AC7 below. They are calculated from the 132,528,706 event causality candidates in Section 3.1. We omit Do et al.'s $Dist$, which is a constant since we set our window size to one.
AC3: Do et al.'s S_{pa}. This is the association measure between arguments and predicates, which is the sum of AC8 and AC9. They are calculated from the 132,528,706 event causality candidates.
AC4: Do et al.'s S_{aa}, which is PMI between arguments. We obtained it in the same way as Filter 5 in the supplementary notes.
AC5: PMI between predicates.
AC6 / AC7: Do et al.'s max / IDF.
AC8: PMI between a cause noun and an effect predicate.
AC9: PMI between a cause predicate and an effect noun.

Table 2: CEA-based association features.

tification in *conduct slash-and-burn agriculture→exacerbate desertification* for instance and consist of CEA-, Wikipedia-, definition-, and web-based features. **CEA-based features** are based on the Cause Effect Association (CEA) measure of Do et al. (2011). It consists of association measures like PMI between arguments (nouns), between arguments and predicates, and between predicates (Table 2). Do et al. used it (along with discourse relations) to extract event causality. **Wikipedia-based features** are the co-occurrence counts and the PMI values between cause and effect nouns calculated using Wikipedia (as of 2013-Sep-19). We also checked whether an Wikipedia article whose title is a cause (effect) noun contains its effect (cause) noun, as detailed in Section D.1 in the supplementary notes. **Definition-based features**, as detailed in Section D.2 in the supplementary notes, resemble the Wikipedia-based features except that the information source is the definition sentences automatically acquired from our 600 million web pages using the method of Hashimoto et al. (2011). **Web-based features** provide association measures between nouns using various window sizes in the 600 million web pages. See Section D.3 for detail. Web-based association measures were obtained from the same database as **AC4** in Table 2.

3.2.4 Base Features

Base features represent the basic properties of event causality like nouns, templates, and their excitation polarities (See Section E in the supplementary notes). For **B3** and **B4**, 500 semantic classes were obtained from our web corpus using the method of Kazama and Torisawa (2008).

3.3 Event Causality Scoring

Using the above features, a classifier[6] classifies each event causality candidate into causality and non-causality. An event causality candidate is given a causality score $CScore$, which is the SVM score (distance from the hyperplane) that is normalized to $[0, 1]$ by the sigmoid function $\frac{1}{1+e^{-x}}$. Each event causality candidate may be given multiple original sentences, since a phrase pair can appear in multiple sentences, in which case it is given more than one SVM score. For such candidates, we give the largest score and keep only one original sentence that corresponds to the largest score.[7] Original sentences are also used for scenario generation, as described below.

4 Future Scenario Generation Method

Our future scenario generation method creates scenarios by chaining event causalities. A naive approach chains two phrase pairs by exact matching. However, this approach would overlook many acceptable scenarios as discussed in Section 1. For example, *global warming worsens→sea temperatures are high* and *sea temperatures rise→vibrio parahaemolyticus fouls (water)* can be chained to constitute an acceptable scenario, but the joint part is not the same string. Note that the two phrases are not simply paraphrases; temperatures may be rising but remain cold, or they may be decreasing even though they remain high.

What characterizes two phrases that can be the joint part of acceptable scenarios? Although we have no definite answer yet, we *name* it the *causal-compatibility* of two phrases and provide its preliminary characterization based on the excitation polarity. Remember that excitatory templates like *cause X* entail that *X*'s function or effect is activated, but inhibitory templates like *lower X* entail that it is suppressed (Section 3.1). Two phrases are *causally-compatible* if they mention the same entity (typically described by a noun) that is predicated by the templates of the *same excitation polarity*. Indeed, both *X rise* and *X are high* are excitatory and hence *sea temperatures are high* and *sea temperatures rise* are causally-compatible.[8]

[6] We used SVM^{light} with the polynominal kernel ($d = 2$), available at `http://svmlight.joachims.org`.

[7] Future work will exploit other original sentences, as suggested by an anonymous reviewer.

[8] Using other knowledge like verb entailment (Hashimoto et al., 2009) can be helpful too, which is further future work.

Scenarios (scs) generated by chaining causally-compatible phrase pairs are scored by $Score(sc)$, which embodies our assumption that an acceptable scenario consists of plausible event causality pairs:

$$Score(sc) = \prod_{cs \in CAUS(sc)} CScore(cs)$$

where $CAUS(sc)$ is a set of event causality pairs that constitutes sc and cs is a member of $CAUS(sc)$. $CScore(cs)$, which is cs's score, was described in Section 3.3.

Our method optionally applies the following two filters to scenarios for better precision: An **original sentence filter** removes a scenario if two event causality pairs that are chained in it are extracted from original sentences between which no word overlap exists other than words constituting causality pairs. In this case, the two event causality pairs tend to be about different topics and constitute an incoherent scenario. A **common argument filter** removes a scenario if a joint part consists of two templates that share no argument in our ⟨argument, template⟩ database, which is compiled from the syntactic dependency data between arguments and templates extracted from our web corpus. Such a scenario tends to be incoherent too.

5 Experiments

5.1 Event Causality Extraction

Next we describe our experiments on event causality extraction and show **(a)** that most of our features are effective and **(b)** that our method outperforms the baselines based on state-of-the-art methods (Do et al., 2011; Hashimoto et al., 2012). Our method achieved 70% precision at 13% recall; we can extract about 69,700 event causality pairs with 70% precision, as described below.

For the **test data**, we randomly sampled 23,650 examples of ⟨event causality candidate, original sentence⟩ among which 3,645 were positive from 2,451,254 event causality candidates extracted from our web corpus (Section 3.1). For the **development data**, we identically collected 11,711 examples among which 1,898 were positive. These datasets were annotated by three annotators (not the authors), who annotated the event causality candidates without looking at the original sentences. The final label was determined by majority vote. The **training data** were created by the annotators through our preliminary experiments and consists of 112,110 among which 9,657 were positive. The Kappa (Fleiss, 1971) of their judgments was 0.67 (substantial agreement (Landis and Koch, 1977)). These three datasets have no overlap in terms of phrase pairs. About nine man-months were required to prepare the data.

Method	Ave. prec. (%)
Proposed	**46.27**
w/o Context features	45.68
w/o Association features	45.66
w/o Semantic relation features	44.44
Base features only	41.29

Table 3: Ablation tests.

Semantic relations	Ave. prec. (%)
All semantic relations (Proposed)	**46.27**
CAUSATION	45.86
CAUSATION and PREVENTION	45.78
None (w/o Semantic relation features)	44.44

Table 4: Ablation tests on semantic relations.

Our evaluation is based on *average precision*;[9] we believe that it is important to *rank* the plausible event causality candidates higher.

5.1.1 Ablation Tests

We evaluated the features of our method by ablation tests. Table 3 shows the results of removing the semantic relation, the context, and the association features from our method. All the feature types are effective and contribute to the performance gain that was about 5% higher than the **Base features only**. **Proposed** achieved 70% precision at 13% recall. We then estimated that, with the precision rate, we can extract 69,700 event causality pairs from the 2,451,254 event causality candidates, among which the estimated number of positive examples is 377,794.

Next we examined whether the semantic relations that do not seem directly relevant to causality like MATERIAL are effective. Table 4 shows that the performance degraded (46.27 → 45.86) when we only used the CAUSATION binary patterns and their entailing and entailed patterns compared to **Proposed**. Even when adding the PREVENTION ("negative CAUSATION") patterns and their entailing and entailed patterns, the performance was still slightly worse than **Proposed**. The performance was even worse when using no semantic relation ("None" in Table 4). Consequently we conclude that not only semantic relations directly relevant

[9] It is obtained by computing the precision for each point in the ranked list where we find a positive sample and averaging all the precision figures (Manning and Schütze, 1999).

Method	Ave. prec. (%)
w/o Wikipedia-based features	**46.52**
Proposed	46.27
w/o definition-based features	46.21
w/o Web-based features	46.15
w/o CEA-based features	45.80

Table 5: Ablation tests on association features.

Method	Ave. prec. (%)
Proposed	**46.27**
Proposed-CEA	45.80
CEA$_{sup}$	21.77
CEA$_{uns}$	16.57

Table 6: Average precision of our proposed methods and baselines using CEA.

to causality like CAUSATION but also those that seem to lack direct relevance to causality like MATERIAL are somewhat effective.

Finally, Table 5 shows the performance drop by removing the Wikipedia-, definition-, web-, and CEA-based features. The CEA-based features were the most effective, while the Wikipedia-based ones slightly degraded the performance.

5.1.2 Comparison to Baseline Methods

We compared our method and two baselines based on Do et al. (2011): **CEA**$_{uns}$ is an unsupervised method that uses CEA to rank event causality candidates, and **CEA**$_{sup}$ is a supervised method using SVM and the CEA features, whose ranking is based on the SVM scores. The baselines are not complete implementations of Do et al.'s method which uses discourse relations identified based on Lin et al. (2010) and exploits them with CEA within an ILP framework. Nonetheless, we believe that this comparison is informative since CEA can be seen as the main component; they achieved a F1 of 41.7% for extracting causal event relations, but with only CEA they still achieved 38.6%.

Table 6 shows the average precision of the compared methods. **Proposed** is our proposed method. **Proposed-CEA** is **Proposed** without the CEA-features and shows their contribution. **Proposed** is the best and the CEA features slightly contribute to the performance, as **Proposed-CEA** indicates. We observed that **CEA**$_{sup}$ and **CEA**$_{uns}$ performed poorly and tended to favor event causality candidates whose phrase pairs were highly relevant to each other but described the contrasts of events rather than event causality (e.g. *build a slow muscle* and *build a fast muscle*) probably because their

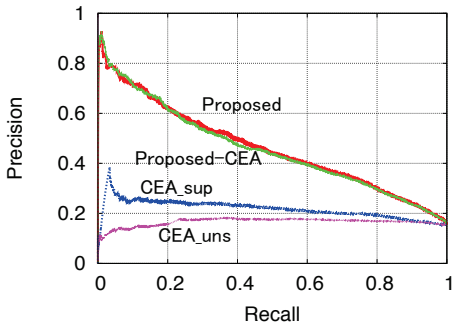


Figure 1: Precision-recall curves of proposed methods and baselines using CEA.

Method	Ave. prec. (%)
Proposed	**49.64**
Cs$_{uns}$	30.38
Cs$_{sup}$	27.49

Table 7: Average precision of our proposed method and baselines using Cs.

main components are PMI values. Figure 1 shows their precision-recall curves.

Next we compared our method with the baselines based on Hashimoto et al. (2012). They developed an automatic excitation template acquisition method that assigns each template an *excitation value* in range $[-1, 1]$ that is positive if the template is excitatory and negative if it is inhibitory. They ranked event causality candidates by $Cs(p_1, p_2) = |s_1| \times |s_2|$, where p_1 and p_2 are the two phrases of event causality candidates, and $|s_1|$ and $|s_2|$ are the absolute excitation values of p_1's and p_2's templates. The baselines are as follows: **Cs**$_{uns}$ is an unsupervised method that uses Cs for ranking, and **Cs**$_{sup}$ is a supervised method using SVM with Cs as the only feature that uses SVM scores for ranking. Note that some event causality candidates were not given excitation values for their templates, since some templates were acquired by manual annotation without Hashimoto et al.'s method. To favor the baselines for fairness, the event causality candidates of the development and test data were restricted to those with excitation values. Since **Cs**$_{sup}$ performed slightly better when using all of the training data in our preliminary experiments, we used all of it.

Table 7 shows the average precision of the compared methods. **Proposed** is our method. Its average precision is different from that in Table 6 due to the difference in test data described above. **Cs**$_{uns}$ and **Cs**$_{sup}$ did not perform well. Many

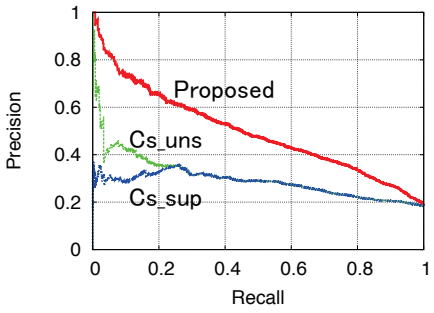


Figure 2: Precision-recall curves of proposed methods and baselines using Cs.

phrase pairs described two events that often happen in parallel but are not event causality (e.g. *reduce the intake of energy* and *increase the energy consumption*) in the highly ranked event causality candidates of **Cs**$_{uns}$ and **Cs**$_{sup}$. Figure 2 shows their precision-recall curves.

Hashimoto et al. (2012) extracted 500,000 event causalities with about 70% precision. However, as described in Section 1, our event causality criteria are different; since they regarded phrase pairs that were not self-contained as event causality (their annotators checked the original sentences of phrase pairs to see if they were event causality), their judgments tended to be more lenient than ours, which explains the performance difference.

In preliminary experiments, since our proposed method's performance degraded when Cs was incorporated, we did not use it in our method.

5.2 Future Scenario Generation

To show that our future scenario generation methods can generate many acceptable scenarios with reasonable precision, we experimentally compared four methods: **Proposed**, our scenario generation method without the two filters, **Proposed+Orig**, our method with the original sentence filter, **Proposed+Orig+Comm**, our method with the original sentence and common argument filters, and **Exact**, a method that chains event causality by exact matching.

Beginning events As the beginning event of a scenario, we extracted nouns that describe social problems (*social problem nouns*, e.g. *deforestation*) from Wikipedia to focus our evaluation on the ability to generate scenarios about them, which is a realistic use-case of scenario generation. We extracted 557 social problem nouns and used the cause phrases of the event causality candidates that consisted of one of the social problem nouns as the scenario's beginning event.

	Two-step	Three-step
Exact	1,000 (44.10)	1,000 (23.50)
Proposed	2,000 (32.25)	2,000 (12.55)
Proposed+Orig	995 (36.28)	602 (17.28)
Proposed+Orig+Comm	708 (38.70)	339 (17.99)

Table 8: Number of scenario samples and their precision (%) in parentheses.

Event causality We applied our event causality extraction method to 2,451,254 candidates (Section 3.1) and culled the top 1,200,000 phrase pairs from them (See Section F in the supplementary notes for examples). Some phrase pairs have the same noun pairs and the same template polarity pairs (e.g. *omit toothbrushing→get a cavity* and *neglect toothbrushing→have a cavity*, where *omit X* and *neglect X* are inhibitory and *get X* and *have X* are excitatory). We removed such phrase pairs except those with the highest $CScore$, and 960,561 phrase pairs remained, from which we generated two- or three-step scenarios that consisted of two or three phrase pairs.

Evaluation samples The numbers of two- and three-step scenarios generated by **Proposed** were 217,836 and 5,288,352, while those of **Exact** were 22,910 and 72,746. We sampled 2,000 from **Proposed**'s two- and three-step scenarios and 1,000 from those of **Exact**. We applied the filters to the sampled scenarios of **Proposed**, and the results were regarded as the sample scenarios of **Proposed+Orig** and **Proposed+Orig+Comm**. Table 8 shows the number and precision of the samples. Note that, for the diversity of the sampled scenarios, our sampling proceeded as follows: **(i)** Randomly sample a beginning event phrase from the generated scenarios. **(ii)** Randomly sample an effect phrase for the beginning event phrase from the scenarios. **(iii)** Regarding the effect phrase as a cause phrase, randomly sample an effect phrase for it, and repeat (iii) up to the specified number of steps (2 or 3). The samples were annotated by three annotators (not the authors), who were instructed to regard a sample as acceptable if each event causality that constitutes it is plausible and the sample as a whole constitutes a single coherent story. Final judgment was made by majority vote. Fleiss' kappa of their judgments was 0.53 (moderate agreement), which is lower than the kappa for the causality judgment. This is probably because

	Two-step	Three-step
Exact	2,085	1,237
Proposed	5,773	0
Proposed+Orig	4,107	0
Proposed+Orig+Comm	3,293	21,153

Table 9: Estimated number of acceptable scenarios with a 70% precision rate.

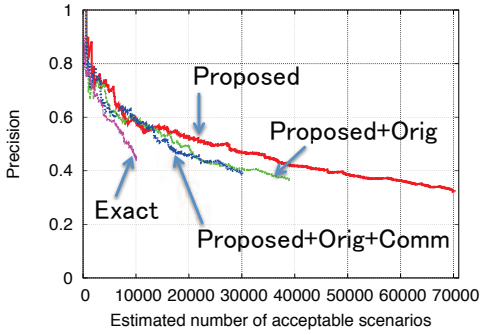


Figure 3: Precision-scenario curves (2-step).

scenario judgment requires careful consideration about various possible futures for which individual annotators tend to draw different conclusions.

Result 1 Table 9 shows the estimated number of acceptable scenarios generated with 70% precision. The estimated number is calculated as the product of the recall at 70% precision and the number of acceptable scenarios in all the generated scenarios, which is estimated by the annotated samples. Figures 3 and 4 show the *precision-scenario curves* for the two- and three-step scenarios, which illustrate how many acceptable scenarios can be generated with what precision. The curve is drawn in the same way as the precision-recall curve except that the X-axis indicates the estimated number of acceptable scenarios. At 70% precision, all of the proposed methods outperformed **Exact** in the two-step setting, and **Proposed+Orig+Comm** outperformed **Exact** in the three-step setting.

Result 2 To evaluate the top-ranked scenarios of **Proposed+Orig+Comm** in the three-step setting with more samples, the annotators labeled 500 samples from the top 50,000 of its output. 341 (68.20%) were acceptable, and the estimated number of acceptable scenarios at a precision rate of 70% and 80% are 26,700 and 5,200 (See Section H in the supplementary notes). The "*terrorist operations*" scenario and the "*slash-and-burn agriculture*" scenario in Section 1 were ranked 16,386th

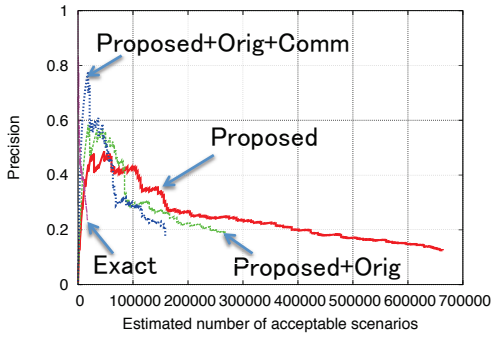


Figure 4: Precision-scenario curves (3-step).

and 21,968th. Next we examined how many of the top 50,000 scenarios were acceptable and *non-trivial*, i.e., found in no page in our input web corpus, using the 341 acceptable samples. A scenario was regarded as non-trivial if its nouns co-occur in no page of the corpus. 22 among the 341 samples were non-trivial. Accordingly, we estimate that we can generate 2,200 ($\frac{50{,}000\times 22}{500}$) acceptable and non-trivial scenarios from the top 50,000. (See Section G in the supplementary notes for examples of the generated scenarios.)

Discussion Scenario *deforestation continues→global warming worsens→sea temperatures rise→vibrio parahaemolyticus fouls (water)* was generated by **Proposed+Orig+Comm**. It is written in no page in our input web corpus, which was crawled in 2007.[10] But we did find a paper Baker-Austin et al. (2013) that observed the emerging vibrio risk in the Baltic sea due to global warming. In a sense, we "predicted" an event observed in 2013 from documents written in 2007, although the scenario was ranked as low as 240,738th.

6 Conclusion

We proposed a supervised method for event causality extraction that exploits semantic relation, context, and association features. We also proposed methods for our new task, future scenario generation. The methods chain event causality by causal-compatibility. We generated non-trivial scenarios with reasonable precision, and "predicted" future events from web documents. Increasing their rank is future work.

[10]The corpus has pages where *global warming*, *sea temperatures*, and *vibrio parahaemolyticus* happen to co-occur. But they are either diaries where the three words appear separately in different topics or lists of arbitrary words.

References

Shuya Abe, Kentaro Inui, and Yuji Matsumoto. 2008. Two-phrased event relation acquisition: Coupling the relation-oriented and argument-oriented approaches. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008)*, pages 1–8.

Susumu Akamine, Daisuke Kawahara, Yoshikiyo Kato, Tetsuji Nakagawa, Yutaka I. Leon-Suematsu, Takuya Kawada, Kentaro Inui, Sadao Kurohashi, and Yutaka Kidawara. 2010. Organizing information on the web to support user judgments on information credibility. In *Proceedings of 2010 4th International Universal Communication Symposium Proceedings (IUCS 2010)*, pages 122–129.

Craig Baker-Austin, Joaquin A. Trinanes, Nick G. H. Taylor, Rachel Hartnell, Anja Siitonen, and Jaime Martinez-Urtaza. 2013. Emerging vibrio risk at high latitudes in response to ocean warming. *Nature Climate Change*, 3:73–77.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of the 48th Annual Meeting of the Association of Computational Linguistics: Human Language Technologies (ACL-08: HLT)*, pages 789–797.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP (ACL-IJCNLP 2009)*, pages 602–610.

Stijn De Saeger, Kentaro Torisawa, Jun'ichi Kazama, Kow Kuroda, and Masaki Murata. 2009. Large scale relation acquisition using class dependent patterns. In *Proceedings of the IEEE International Conference on Data Mining (ICDM 2009)*, pages 764–769.

Stijn De Saeger, Kentaro Torisawa, Masaaki Tsuchida, Jun'ichi Kazama, Chikara Hashimoto, Ichiro Yamada, Jong-Hoon Oh, István Varga, and Yulan Yan. 2011. Relation acquisition using word classes and partial patterns. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*, pages 825–835.

Quang Xuan Do, Yee Seng Chan, and Dan Roth. 2011. Minimally supervised event causality identification. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*, pages 294–303.

Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382.

Chikara Hashimoto, Kentaro Torisawa, Kow Kuroda, Masaki Murata, and Jun'ichi Kazama. 2009. Large-scale verb entailment acquisition from the web. In *Proceedings of EMNLP 2009: Conference on Empirical Methods in Natural Language Processing*, pages 1172–1181.

Chikara Hashimoto, Kentaro Torisawa, Stijn De Saeger, Jun'ichi Kazama, and Sadao Kurohashi. 2011. Extracting paraphrases from definition sentences on the web. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1087–1097.

Chikara Hashimoto, Kentaro Torisawa, Stijn De Saeger, Jong-Hoon Oh, and Jun'ichi Kazama. 2012. Excitatory or inhibitory: A new semantic orientation extracts contradiction and causality from the web. In *Proceedings of EMNLP-CoNLL 2012: Conference on Empirical Methods in Natural Language Processing and Natural Language Learning*, pages 619–630.

Jun'ichi Kazama and Kentaro Torisawa. 2008. Inducing gazetteers for named entity recognition by large-scale clustering of dependency relations. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-08: HLT)*, pages 407–415.

Julien Kloetzer, Stijn De Saeger, Kentaro Torisawa, Chikara Hashimoto, Jong-Hoon Oh, and Kiyonori Ohtake. 2013a. Two-stage method for large-scale acquisition of contradiction pattern pairs using entailment. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, pages 693–703.

Julien Kloetzer, Kentaro Torisawa, Stijn De Saeger, Motoki Sano, Chikara Hashimoto, and Jun Gotoh. 2013b. Large-scale acquisition of entailment pattern pairs. In *Information Processing Society of Japan (IPSJ) Kansai-Branch Convention 2013*.

J. Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2010. A pdtb-styled end-to-end discourse parser. Technical report, School of Computing, National University of Singapore.

Chris Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

Jong-Hoon Oh, Kentaro Torisawa, Chikara Hashimoto, Motoki Sano, Stijn De Saeger, and Kiyonori Ohtake. 2013. Why-question answering using intra- and inter-sentential causal relations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pages 1733–1743.

Kira Radinsky and Eric Horvitz. 2013. Mining the web to predict future events. In *Proceedings of Sixth ACM International Conference on Web Search and Data Mining (WSDM 2013)*, pages 255–264.

Kira Radinsky, Sagie Davidovich, and Shaul Markovitch. 2012. Learning causality for news events prediction. In *Proceedings of International World Wide Web Conference 2012 (WWW 2012)*, pages 909–918.

Mehwish Riaz and Roxana Girju. 2010. Another look at causality: Discovering scenario-specific contingency relationships with no supervision. In *2010 IEEE Fourth International Conference on Semantic Computing*, pages 361–368.

Peter Schwartz. 1991. *The Art of the Long View*. Doubleday.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2005. Extracting semantic orientation of words using spin model. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pages 133–140.

Shohei Tanaka, Naoaki Okazaki, and Mitsuru Ishizuka. 2012. Acquiring and generalizing causal inference rules from deverbal noun constructions. In *Proceedings of 24th International Conference on Computational Linguistics (COLING 2012)*, pages 1209–1218.

Masahiro Tanaka, Stijn De Saeger, Kiyonori Ohtake, Chikara Hashimoto, Makoto Hijiya, Hideaki Fujii, and Kentaro Torisawa. 2013. WISDOM2013: A large-scale web information analysis system. In *Companion Volume of the Proceedings of the 6th International Joint Conference on Natural Language Processing (IJCNLP 2013) (Demo Track)*, pages 45–48.

Kentaro Torisawa, Stijn de Saeger, Jun'ichi Kazama, Asuka Sumida, Daisuke Noguchi, Yasunari Kakizawa, Masaki Murata, Kow Kuroda, and Ichiro Yamada. 2010. Organizing the web's information explosion to discover unknown unknowns. *New Generation Computing (Special Issue on Information Explosion)*, 28(3):217–236.

Kentaro Torisawa. 2006. Acquiring inference rules with temporal constraints by using japanese coordinated sentences and noun-verb co-occurrences. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL (HLT-NAACL2006)*, pages 57–64.

Masaaki Tsuchida, Kentaro Torisawa, Stijn De Saeger, Jong Hoon Oh, Jun'ichi Kazama, Chikara Hashimoto, and Hayato Ohwada. 2011. Toward finding semantic relations not written in a single sentence: An inference method using auto-discovered rules. In *Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP 2011)*, pages 902–910.

István Varga, Motoki Sano, Kentaro Torisawa, Chikara Hashimoto, Kiyonori Ohtake, Takao Kawai, Jong-Hoon Oh, and Stijn De Saeger. 2013. Aid is out there: Looking for help from tweets during a large scale disaster. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pages 1619–1629.

Naoki Yoshinaga and Masaru Kitsuregawa. 2009. Polynomial to linear: Efficient classification with conjunctive features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP 2009)*, pages 542–1551.

Cross-narrative temporal ordering of medical events

Preethi Raghavan*, **Eric Fosler-Lussier***, **Noémie Elhadad**† **and Albert M. Lai***
*The Ohio State University, Columbus, Ohio
†Columbia University, New York, NY
{raghavap, fosler}@cse.ohio-state.edu
noemie.elhadad@columbia.edu, albert.lai@osumc.edu

Abstract

Cross-narrative temporal ordering of medical events is essential to the task of generating a comprehensive timeline over a patient's history. We address the problem of aligning multiple medical event sequences, corresponding to different clinical narratives, comparing the following approaches: (1) A novel weighted finite state transducer representation of medical event sequences that enables composition and search for decoding, and (2) Dynamic programming with iterative pairwise alignment of multiple sequences using global and local alignment algorithms. The cross-narrative coreference and temporal relation weights used in both these approaches are learned from a corpus of clinical narratives. We present results using both approaches and observe that the finite state transducer approach performs performs significantly better than the dynamic programming one by 6.8% for the problem of multiple-sequence alignment.

1 Introduction

Discourse structure, logical flow of sentences, and context play a large part in ordering medical events based on temporal relations within a clinical narrative. However, cross-narrative temporal relation ordering is a challenging task as it is difficult to learn temporal relations among medical events which are not part of the logically coherent discourse of a single narrative. Resolving cross-narrative temporal relationships between medical events is essential to the task of generating an event timeline from across unstructured clinical narratives such as admission notes, radiology reports, history and physical reports and discharge summaries. Such a timeline has multiple applications in clinical trial recruitment (Luo et al., 2011), medical document summarization (Bramsen et al., 2006, Reichert et al., 2010) and clinical decision making (Demner-Fushman et al., 2009).

Given multiple temporally ordered medical event sequences generated from each clinical narrative in a patient record, how can we combine the events to create a timeline across all the narratives? The tendency to copy-paste text and summarize past information in newly generated clinical narratives leads to multiple mentions of the same medical event across narratives (Cohen et al., 2013). These cross-narrative coreferences act as important anchors for reasoning with information across narratives. We leverage cross-narrative coreference information along with confident cross-narrative temporal relation predictions and learn to align and temporally order medical event sequences across longitudinal clinical narratives. We model the problem as a sequence alignment task and propose solving this using two approaches. First, we use weighted finite state machines to represent medical events sequences, thus enabling composition and search to obtain the most probable combined sequence of medical events. As a contrast, we adapt dynamic programming algorithms (Needleman et al., 1970, Smith and Waterman, 1981) used to produce global and local alignments for aligning sequences of medical events across narratives. We also compare the proposed methods with an Integer Linear Programming (ILP) based method for timeline construction (Do et al., 2012). The cross-narrative coreference and temporal relation scores used in both these approaches are learned from a corpus of patient narratives from The Ohio State University Wexner Medical Center.

The main contribution of this paper is a general framework that allows aligning multiple event sequences using cascaded weighted finite state transducers (WFSTs) with the help of efficient composition and decoding. Moreover, we demonstrate that this method can be used for more accurate multiple sequence alignment when compared to

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 998–1008,
Baltimore, Maryland, USA, June 23-25 2014.

dynamic programming or other ILP-based methods proposed in literature.

2 Related Work

In the areas of summarization and text-to-text generation, there has been prior work on several ordering strategies to order pieces of information extracted from different input documents (Barzilay et al., 2002, Lapata, 2003, Bollegala et al., 2010). In this paper, we focus on temporal ordering of information, as discussed next.

Recent state-of-the art research has focused on the problem of temporal relation learning within the same document, and in many cases within the same sentence (Mani et al., 2006, Verhagen et al., 2009, Lapata and Lascarides, 2011). Chambers and Jurafsky (2009) describe a process to induce a partially ordered set of events related by a common protagonist by using an unsupervised distributional method to learn relations between events sharing coreferring arguments, followed by temporal classification to induce partial order. The task was carried out on the Timebank newswire corpus, but was limited to an intra-document setting. More recently, (Do et al., 2012) proposed an ILP-based method to combine the outputs of an event-interval and an event-event classifier for timeline construction on the ACE 2005 corpus. However, this approach is also restricted to events within documents and requires annotations for event intervals. We empirically compare our methods for timeline creation from longitudinal clinical narratives to such an ILP-based approach in Section 7. While a lot of this work has been done in the news domain, there is also some recent work in rule-based algorithms (Zhou et al., 2006) and machine learning (Roberts et al., 2008) applied to temporal relations between medical events in clinical text. Clinical narratives are written in a distinct sub-language with domain specific terminology and temporal characteristics, making them markedly different from newswire text.

There is limited prior work in learning relations across documents. Ji and Grishman (2008) extended the one sense per discourse idea (Yarowsky, 1995) to multiple topically related documents and propagate consistent event arguments across sentences and documents. Barzilay and McKeown (2005) propose a text-to-text generation technique for synthesizing common information across documents using sentence fusion. This involves multisequence dependency tree alignment to identify phrases conveying similar information and statistical generation to combine common phrases into a sentence. Along with syntactic features, they combine knowledge from resources like WordNet to find similar sentences. In case of clinical narratives and medical event alignment, the objective is to identify a unique sequence of temporally ordered medical events from across longitudinal clinical data.

To the best of our knowledge, there is no prior work on cross-document alignment of event sequences. Multiple sequence alignment is a problem that arises in a variety of domains including gene/protein alignments in bioinformatics (Notredame, 2002), word alignments in machine translation (Kumar and Byrne, 2003), and sentence alignments for summarization (Lacatusu et al., 2004). Dynamic programming algorithms have been popularly leveraged to produce pairwise and global genetic alignments, where edit distance based metrics are used to compute the cost of insertions, deletions and substitutions. We use dynamic programming to compute the best alignment, given the temporal and coreference information between medical events across these sequences. More importantly, we propose a cascaded WFST-based framework for cross-document temporal ordering of medical event sequences. Composition and search operations can be used to build a single transducer that integrates these components, directly mapping from input states to desired outputs, and obtain the best alignment (Mohri et al., 2000). In natural language processing, WFSTs have seen varied applications in machine translation (Kumar and Byrne, 2003), morphology (Sproat, 2006), named entity recognition (Krstev et al., 2011) and biological sequence alignment / generation (Whelan et al., 2010) among others. We demonstrate that the WFST-based approach outperforms popularly used dynamic programming algorithms for multiple sequence alignment.

3 Problem Description

Medical events are temporally-associated concepts in clinical text that describe a medical condition affecting the patient's health, or procedures performed on a patient. We represent medical events by splitting each event into a start and a stop. When there is insufficient information to discern the start or stop of an event, it is represented as a single concept. If only the start is known then the stop is set to $+\infty$, whereas when only the stop is known , the start is set to the date of birth of the

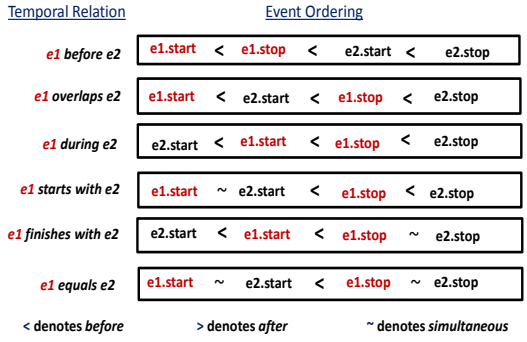


Figure 1: Medical event start / stop representation mapped to Allen's temporal relations (Allen, 1981). Temporal ordering of event starts and stops using $\{before, after, simultaenous\}$ (shown on the right) allows us learn temporal relations between the medical events (shown on the left). $e1_{start} = e2_{start}$ and $e1_{stop} = e2_{stop}$, when $e1$ and $e2$ corefer.

patient.[1] Often, for chronic ailments like *hypertension*, we would only associate a start with the medical event and set the stop to $+\infty$. The start of *hypertension* may be associated with the temporal expression *history of* in the narrative. This, when considered along with the admission date, allows us to relatively order *hypertension* with respect to other medical events. A medical event occurrence like *chest pain* may be associated with a start and a stop, where the start may be determined by the mention of "patient was complaining of chest pain *yesterday*" in the narrative text. Further, the narrative may state that "he *continued* to have chest pain *on admission*, but *currently* he is chest pain free"; this may be used to infer the relative stop of *chest pain*. Medical events may also be instantaneous, for e.g., *injected with antibiotic*. Such events are represented with the start and stop as being the same. Temporal relations exist between the start and stop of events as shown in Figure 1. Learning temporal relations *before, after* and *simultaneous* between the medical event starts and stops corresponds to learning all of Allen's temporal relations (Allen, 1981) between the medical events. Following our previous work (Raghavan et al., 2012c), such a representation allows us to temporally order the event starts and stops within each clinical narrative by learning to rank them in relative order of time. The problem definition is as follows:

[1]Patient date of birth, admission/ discharge date are usually available in the metadata associated with a clinical narrative.

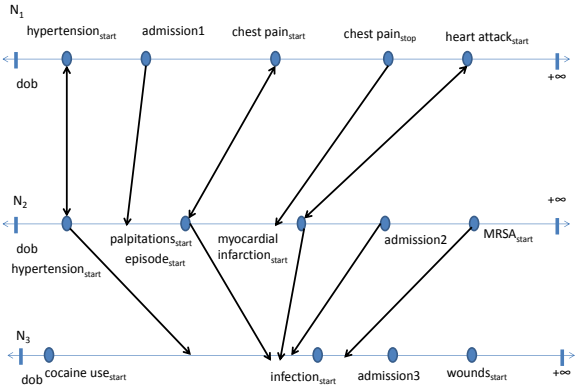


Figure 2: Given temporally ordered medical event sequences, N_1, N_2, N_3, we address the task of combining events across these sequences by merging or ordering them to create a single comprehensive timeline.

Input: Sequences of temporally ordered medical event starts and stops. This corresponds to N_1, N_2, and N_3 in Figure 2. Each sequence corresponds to a clinical narrative. The total number of sequences correspond to the number of clinical narratives for a patient.

Problem: Combine medical events across these sequences to generate a timeline i.e., a single comprehensive sequence of medical events over all clinical narratives of the patient.

Expected Output: In the example shown in Figure 2, the output would be as follows: Timeline (N_1, N_2, N_3)= $\{$cocaine use$_{start}$ $<$ hypertension$_{start}$ $=$ hypertension$_{start}$ $<$ admission1 $<$ chest pain$_{start}$ $\sim$ palpitations$_{start}$ $<$ chest pain$_{stop}$ $<$ heart attack$_{start}$ $=$ myocardial infarction$_{start}$ $<$ admission2 $<$ infection$_{start}$ $<$ MRSA$_{start}$ $<$ admission3 $<$ wounds$_{start}\}$.

The goal of multiple sequence alignment is to find an alignment that maximizes some overall alignment score. Thus, in order to align event sequences, we need to compute scores corresponding to cross-narrative medical event coreference resolution and cross-narrative temporal relations.

4 Cross-Narrative Coreference Resolution and Temporal Relation Learning

The first approach to learning a temporal ordering of medical events across all clinical narratives is to consider all pairs of events across all narratives and learn to classify them as sharing one of Allen's temporal relations (Allen, 1981) using a single learning model. Alternatively, a ranking ap-

proach, similar to the one used to generate intra-narrative temporal ordering, can also be extended to the cross-narrative case. However, the features related to narrative structure and relative and implicit temporal expressions used for temporal ordering within a clinical narrative may not be applicable across narratives. For instance, a history and physical report may have sections like "past medical history", "history of present illness", "assessment and plan", and a certain logical pattern to the flow of text within and across these sections. Further, temporal cues like "thereafter", "subsequently", follow from the context around an event mention. The absence of such features in the cross-narrative case does not allow such a model to generate accurate temporal relation predictions.

Thus, for use in our sequence alignment models, we learn two independent classifiers for medical event coreference and temporal relation learning across narratives. We train a classifier to resolve cross-narrative coreferences by extracting semantic and temporal relatedness feature sets for each pair of medical concepts. Extracting these feature sets helps us train a classifier to predict medical event coreferences (Raghavan et al., 2012a). Another classifier is then trained to classify pairs of medical event starts and stops across narratives as sharing temporal relations {before, after, overlaps}. The learned cross-narrative coreference predictions can then be used along with confident temporal relation predictions to derive a joint probability to enable cross-narrative temporal ordering.

5 Narrative Sequence Alignment for Cross-narrative Temporal Ordering

Sequence alignment algorithms have been developed and popularly used in bioinformatics. However, multiple sequence alignment (MSA) has been shown to be NP complete (Wang and Jiang, 1994) and various heuristic algorithms have been proposed to solve this problem (Notredame, 2002). We propose a novel WFST-based representation that enables accurate decoding for MSA when compared to popularly used dynamic programming algorithms (Needleman et al., 1970, Smith and Waterman, 1981) or other state of the art methods (Do et al., 2012).

In the problem of aligning events across multiple narrative sequences, we want to align temporally ordered medical events corresponding to clinical narratives of a patient. Unlike problems in biological sequence alignment where the symbols to be aligned across sequences are restricted to a fixed set, our symbol set is not fixed or certain because the symbols correspond to medical events in clinical narratives. Moreover, we cannot have fixed scores for symbol transformations since our transformations correspond to coreference and temporal relations between the medical events across sequences. The computation of these scores is described next.

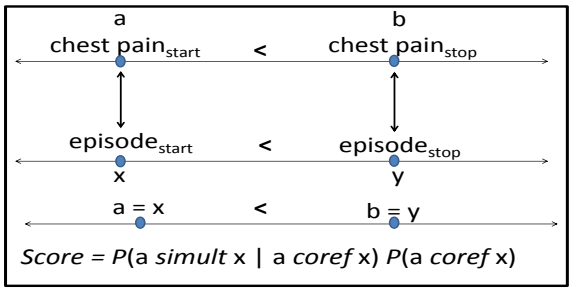


Figure 3: Score computation for aligning events across temporally ordered event sequences chest pain$_{start}$ = episode$_{start}$ < chest pain$_{stop}$ = episode$_{stop}$, where events across the sequences occur simultaneously and corefer.

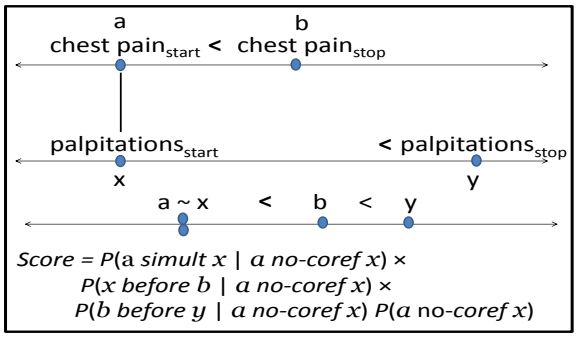


Figure 4: Score computation for aligning events across temporally ordered event sequences chest pain$_{start}$ ∼ palpitations$_{stop}$ < chest pain$_{stop}$ < palpitations$_{stop}$, where some events across the sequences occur simultaneously but do not corefer.

5.1 Scoring Scheme

Let us assume a, b are medical events in the first clinical narrative and have been temporally ordered so $a < b$. Similarly, x, y are medical events in the second clinical narrative such that $x < y$. There exists a match or an alignment between a pair of medical events, across the sequences, in the following cases:

1. If the medical events are simultaneous and coreferring, denoted as $a = x$.
2. If the medical events are simultaneous and non-coreferring, denoted as $a \sim x$.

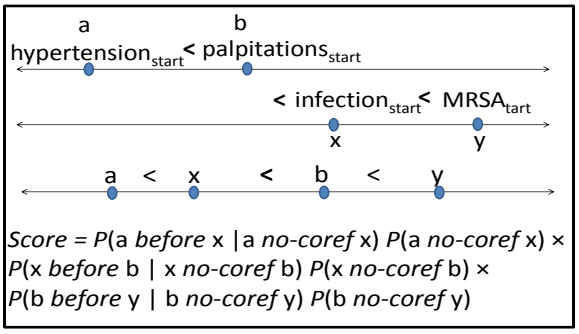


Figure 5: Score computation for aligning events across temporally ordered event sequences hypertension$_{start}$ $<$ palpitations$_{start}$ $<$ infection$_{start}$ $<$ MRSA$_{start}$, where events across the sequences do not occur simultaneously and do not corefer.

3. If the a medical event from one sequence is before a medical event from another sequence, denoted as $a < x$.
4. If the a medical event from one sequence is after a medical event from another sequence, denoted as $a > x$.

We now illustrate how the scores for candidate aligned sequences are computed using the learned cross-narrative coreference and temporal probabilities for the following three scenarios:

- The medical events across sequences are simultaneous and corefer as illustrated in Figure 3. The joint score considers the probability of event temporal relations *simultaneous* conditioned on *coreference*.
- Some medical events across sequences are simultaneous but do not corefer as illustrated in Figure 4. Here, the joint score considers the joint probability of temporal relations *simultaneous* or *before* and *no-coreference*.
- The medical events across sequences are not simultaneous and do not corefer as illustrated in Figure 5. In this case, the joint score considers the probability of the temporal relation *before* and *no coreference*.

Thus, the coreference and temporal relation scores can be leveraged for aligning sequences of medical events. These scores are used in both the WFST-based representation and decoding, as well as for dynamic programming.

5.2 Alignment using a Weighted Finite State Representation

A weighted finite-state transducer (WFST) is an automaton in which each transition between states is associated with an input symbol, an output symbol, and a weight (Mohri et al., 2005). WFSTs can be used to efficiently represent and combine sequences of medical events based coreference and temporal relation information. The WFST representation gives us the ability to talk about the global joint probability derived from coreference and temporal relation scores described in Section 5.1. It allows us to build a weighted lattice of sequences that can be searched for the most probable sequence of medical events from across all clinical narratives of a patient. We use unweighted FSAs to represent the input described in Section 3, i.e. temporally ordered sequences of medical events corresponding to clinical narratives. This corresponds to N_1 and N_2 in Figure 6.

Based on whether we want to align the sequences purely based on coreference scores or both coreference and temporal relation scores, the arc weights for the WFST can be determined. M_{12}^{c} is a WFST that maps input symbols from N_1 to output symbols in N_2 and is weighted by the probability of coreference or no-coreference between medical events across N_1 and N_2. The representation in WFST M_{12}^{c+t} shown in Figure 7 allows us to align N_1 and N_2 based on both coreference as well as temporal relation probabilities. The WFST has ϵ transitions to accommodate insertion and deletion of medical events when combining the sequences. Deletions correspond to the case when an event in the first sequence does not map to any event in the second sequence; similarly insertions correspond to the case where an event in the second sequence does not map to any event in the first sequence. The WFST composition operation allows the outputs of one WFST to be fed to the inputs of a second WFST or FSA. Thus, we build our final machine by composing the three sub-machines as,

$$D = N_1 \circ M_{12}^{i} \circ N_2. \quad (1)$$

where $i = c$ or $i = c + t$. This gives us a combined weighted graph by mapping the output symbols of the first medical event sequence to the input symbols of the second medical event sequence. The scores on the decoding graph are derived from only the coreference probabilities if $i = c$ and both coreference and temporal relation probabilities if $i = c + t$.

In the medical event sequence alignment problem, we want to align multiple sequences of medical events that correspond to multiple clinical narratives of a patient. Since we want to now combine

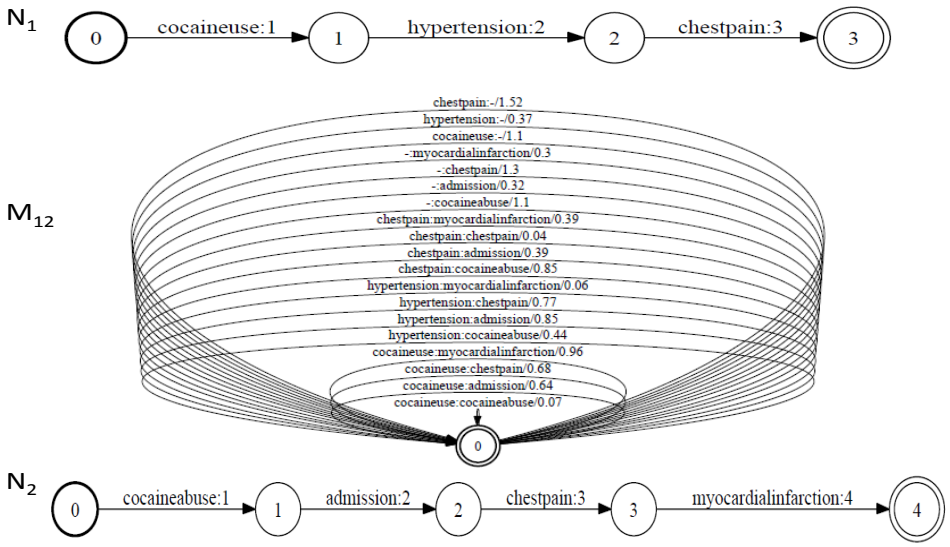


Figure 6: N_1 and N_2 are medical event sequences represented using FSAs. M^c_{12} maps medical events across N_1 and N_2 and is weighted only by the probability of coreference between events across N_1 and N_2.

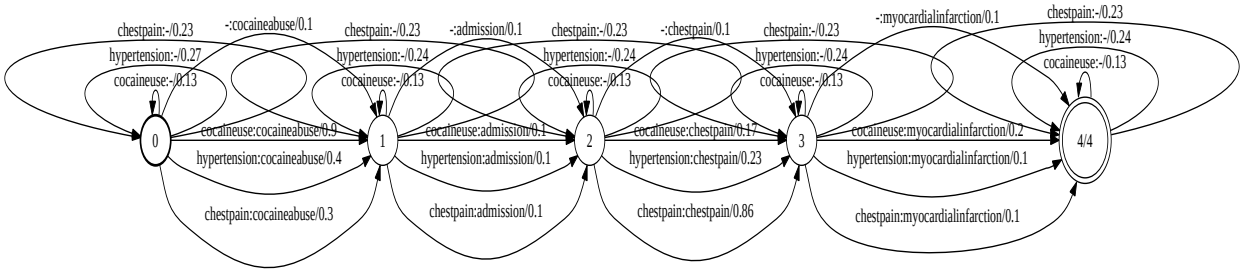


Figure 7: M^{c+t}_{12} is a WFST representation used for mapping medical events between N_1 and N_2 (from Figure 2) and is weighted by both the coreference and temporal relation probabilities

all narrative chains belonging to the same patient, the composition cascade to build the final combined sequence will be as,

$$D_f = N_1 \circ M^i_{12} \circ N_2 \circ M^i_{23} \circ N_3 \circ M^i_{34} ... \circ N_n \quad (2)$$

where $i = c$ or $i = c + t$ and n is the number of medical event sequences corresponding to clinical narratives for a patient. During composition we retain intermediate paths like M^i_{23} utilizing the ability to do lazy composition (Mohri and Pereira, 1998) in order to facilitate beam search through the multi-alignment. The best hypothesis corresponds to the highest scoring path which can be obtained using shortest path algorithms like Djikstra's algorithm. The best path corresponds to the best alignment across all medical event sequences based on the joint probability of cross-narrative medical event coreferences and temporal relations across the narrative sequences.

The complexity of decoding increases exponentially with the number of narrative sequences in the composition, and exact decoding becomes infeasible. One solution to this problem is to do the alignment greedily pairwise, starting from the most recent medical event sequences, finding the best path, and iteratively moving on to the next sequence, and proceeding until the oldest medical event sequence. The disadvantage of such a method is that it does not take into account constraints between medical events across multiple event sequences and may lead to a less accurate solution.

An alternative method is to use lazy composition to perform more efficient composition as it allows practical memory usage. We also use beam search to make for an efficient approximation to the best-path computation (Mohri et al., 2005). This allows accommodating constraints from across multiple sequences and generates a more accurate best path. Thus, this method generates more accurate alignments when we have more than two sequences to be aligned.

For instance, instance say $a, b \in N_1, x, y \in N_2$, and $m, n \in N_3$ are temporally medical event sequences corresponding to narratives N_1, N_2 and N_3. Based on the learned pairwise temporal relations, if we have the following constraints $a < x$, $m > x$, $m < a$. Aligning N_1 and N_2 greedily pairwise may give us the best combined sequence as $a, x, b, y \in N_{12}$. Now in aligning N_{12} with N_3, we won't be able to accommodate $m > x$ and $m < a$. However, performing a beam search over the composed WFST in equation 2 allows us to accommodate such constraints across multiple sequences. The complexity of composing two transducers is $O(V_1 V_2 D_1 (logD_2 + M_2))$ where each edge from the first sequence matches every edge in the second sequence and V_i is the number of states, D_i is the maximum out-degree and M_i maximum multiplicity for the i^{th} FST (Mohri et al., 2005).

We also use popular dynamic programming algorithms (Needleman et al., 1970, Smith and Waterman, 1981) for sequence alignment of medical events across narratives and compare it to the WFST-based representation and decoding.

5.3 Pairwise Alignment using Dynamic Programming

As a contrast, we adapt two dynamic programming algorithms for sequence alignment: global alignment using the Needleman Wunsch algorithm (NW) (Needleman et al., 1970) and local alignment using the Smith-Waterman algorithm (SW) (Smith and Waterman, 1981). NW allows us to align all events in one sequence with all events in another sequence. A drawback of NW is that short and highly similar sequences maybe missed because they get overweighted by the rest of the sequence. NW is suitable when the two sequences are of similar length with significant degree of similarity throughout. On the other hand, SW gives the longest sub-sequence pair that yields maximum degree of similarity between the two original sequences. It does not force all events in a sequence to align with another sequence. SW is useful in aligning sequences that differ in length and have short patches of similarity. The time complexity of these methods for sequences of length m and n are $O(mn)$.

The scoring scheme described earlier is used to update the scoring matrix for dynamic programming. In order to accommodate the temporal relations before and after, we insert a null symbol after every medical event in each sequence in the scoring matrix. A vertical or horizontal gap arises when cases 1, 2, 3 and 4 in Section 5.1 mentioned above are not true. If the medical events are not simultaneous, not before or not after, the medical events will not align. Thus, the value of each cell in the scoring matrix is determined by computing the maximum score at each position $C(i, j)$ as,

$$\max\{(C(i-1, j-1) + S_{ij}), (C(i, j-1) + w), (C(i-1, j) + w)\} \quad (3)$$

where, $S_{ij} = \max\{P(i = j), P(i < j), P(i > j)\}$, and $w = \max\{(1 - P(i = j)), (1 - P(i < j)), (1 - P(i > j))\}$. Here, $C(i-1, j-1)$ corresponds to a match, whereas $C(i, j-1)$ and $C(i-1, j)$ correspond to a gaps in sequence one and two.

In case of the SW algorithm, the negative scoring matrix cells are set to zero, thus making the positively scoring local alignments visible. Backtracking starts at the highest scoring matrix cell and proceeds until a cell with score zero is encountered, yielding the highest scoring local alignment.

The time and space complexity grows exponentially with the number of sequences to be aligned and finding the global optimum has been shown to be a NP-complete problem. The time complexity of aligning N sequences of length L is $O(2^N L^N)$ (Wang and Jiang, 1994). Thus, for MSA using dynamic programming, we use a heuristic method where we combine pairwise alignments iteratively starting with the latest narrative and progressing towards the oldest narrative.

6 Experiments and Evaluation

Corpus Description. The corpus consists of a dataset of clinical narratives obtained from the [redacted] medical center. The corpus has a total of 2060 patients, and 100704 clinical narratives. We gathered a gold standard set of seven patients (80 clinical narratives overall) with manual annotation of all medical events mentioned in the narratives, coreferences, and medical event sequence information. The annotation agreement across annotators is high, with 89.5% agreement corresponding to inter-annotator Cohen's kappa statistic of 0.86 (Raghavan et al., 2012b). The types of clinical narratives included 27 discharge summaries, 30 history and physical reports, 15 radiology reports and 8 pathology reports. The distribution of the number of medical event sequences and unique medical events across patients is shown in Table 1. The annotated dataset is used to cross-validate and train our coreference and temporal relation learning models and to evaluate our cross-narrative medical event timeline.

	p1	p2	p3	p4	p5	p6	p7	
No. of Narrative Sequences	5	9	20	13	8	10	15	
No. of Medical events	68	90	119	82	79	72	95	
	% Accuracy							% Avg.
WFST-framework (lazy composition and beam search)[c+t]	**76.1**	**73.2**	**81.2**	**83.5**	**76.4**	**82.5**	**79.7**	78.9
WFST-framework (Iterative pairwise)[c+t]	70.4	67.1	73.5	74.1	61.8	75.5	62.9	69.3
Smith Waterman (Iterative pairwise)[c+t]	71.2	69.7	75.5	75.6	66.3	77.4	68.3	72.1
Needleman-Wunsch (Iterative pairwise)[c+t]	68.1	66.3	72.1	74.4	61.1	75.5	63.6	68.7
WFST-framework (lazy composition and beam search)[c]	**68.5**	**65.3**	**72.3**	**74.4**	**67.2**	**71.3**	**69.1**	69.7
WFST-framework (Iterative pairwise)[c]	61.2	63.3	61.9	60.4	59.8	64.8	60.5	61.7
Smith Waterman (Iterative pairwise)[c]	60.3	63.7	68.2	62.3	58.6	66.7	60.2	62.8
Needleman-Wunsch (Iterative pairwise)[c]	56.6	60.1	59.3	65.6	54.7	63.1	58.2	59.6

Table 1: The distribution of medical events across narrative sequences and sequences across patients and multiple sequence alignment results for the WFST-based framework, and dynamic programming using just coreference scores [c] and using coreference as well as temporal relation scores [c+t].

Evaluation Metric. For each patient and each method (WFST or dynamic programming), the output timeline to evaluate is the highest scoring candidate hypothesis derived as described above. Accuracy of the timeline is calculated as the number of transformations required to obtain the reference sequence in the annotated gold-standard from the one generated by our system. Transformations are measured in terms of the minimum edit distance, insertions, deletions, and substitutions of medical events.

Experiments and Results. We first temporally order medical events within each clinical narrative by learning to rank them in relative order of occurence as described in our previous work (Raghavan et al., 2012c). The overall accuracy of ranking medical events using leave-one-out cross validation is 82.1%. The resulting medical event sequences serve as the input to the problem of cross-narrative sequence alignment.

The cross-narrative coreference and temporal relation pairwise classification models described in Section 4 are trained using a Maximum entropy classifier. The coreference resolution performs with 71.5% precision and 82.3% recall. The temporal relation classifier performs with 60.2% precision and 76.3% recall. The learned pairwise coreference and temporal relation probabilities are now used to derive the score for the WFST and dynamic programming approaches.

WFST representation and decoding. We build finite-state machines using the open source OpenFST library.[2] We use a tropical semi-ring weighted using the negative log-likelihood of the computed scores. OpenFST provides tools that can search for the highest scoring sequences accepted by the machine, and can sample from high-scoring sequences probabilistically, by treating the scores of each transition within the machine as a negative log probability. The decoding process to compute the most likely combined medical event sequence can be defined as searching for the best path in the combined graph representation (Equation 2). The best path is the one that minimizes the total weight on a path (since the arcs are negative log probabilities). In searching for the best path, the beam size is set to 5. The accuracy of the WFST-based representation and beam search across all sequences using the coreference and temporal relation scores to obtain the combined aligned sequence is 78.9%.

Dynamic Programming. We use the NW and SW algorithms described in Section 5.3 to produce local and global alignments respectively. We use the scoring scheme described in Section 5.1 to update the cost matrix for dynamic programming and implement the algorithms as described in Section 5.3. The overall accuracy of sequence alignment with both coreference and temporal relation scores using NW is 68.7% whereas SW gives an accuracy of 72.1%. In case of aligning just two sequences, both methods yield the same results. The accuracy of cross-narrative MSA for each patient, for each method, using cross validation, is shown in Table 1. Results indicate that the WFST-based method outperforms the dynamic programming approach for multi-sequence alignment (statistical significance $p<0.05$). Morever, the results using both coreference and temporal realtion scores for alignment outperform using only coreference scores for alignment using all approaches. This indicates that cross-narrative temporal relations are important for accurately aligning medical event sequences across narratives.

7 Discussion

We propose and evaluate different approaches to multiple sequence alignment of medical events.

[2] www.openfst.org

Approaches to multi-alignment. We address the problem of aligning medical event sequences using a novel WFST-based framework and empirically demonstrate that it outperforms pairwise progressive alignment using dynamic programming. This is mainly because the WFST-based allows us to consider temporal constraints from across multiple sequences when performing the alignment.

Moreover, it also outperforms the integer linear programming (ILP) method for timeline construction proposed in (Do et al., 2012). We implemented the proposed method that also allows combining the output of classifiers subject to some constraints. We derive intervals from event starts and stops and learn two perceptron classifiers for classifying the temporal relations between events and assigning events to intervals. The classifier probabilities are then used to solve the optimization problem using the lpsolve solver.[3] We also use intra-document coreference information to resolve coreference before performing the global optimization. We observe that in case of MSA, the optimal solution using ILP is still intractable as the number of constraints increases exponentially with the number of sequences. Aligning pairwise iteratively gives us an overall average accuracy of 68.2% similar to dynamic programming. While this is comparable to the dynamic programming performance, the WFST-based method significantly outperforms this in case of multi-alignments for cross-narrative temporal ordering.

Performance and error analysis. We perform multi-alignments over medical event sequences for a patient, where each sequence corresponds to temporally ordered medical events in a clinical narrative generated using the ranking model described in (Raghavan et al., 2012c). The accuracy of intra-narrative temporal ordering is 82.1%. The errors in performing this intra-narrative ordering may propagate to the cross-narrative model resulting in reduced accuracy. This may be addressed by considering n-best temporally ordered medical event sequences, generated by the ranking process, and aligning the n-best sequences using the WFST-based framework. This could be feasible as, practically, the WFST-based method for multi-alignment takes only a few secs to align a pair of medical event sequences with average length 40.

The accuracy of alignments across multiple medical event sequences is also affected by the error induced by the coreference and temporal relation scores. Often, insufficient temporal cues leads to misclassification of events incorrectly as sharing the "simultaneous" temporal relation and often as coreferring. This induces errors in the score calculation and hence the alignments. Better methods to address the challenging problem of cross-document temporal relation learning, perhaps with the help of structured data from the patient record, could improve the accuracy of alignments.

There is no clear trend with respect to the number of medical events and narratives for a patient (Table 1.), and the alignment accuracy. In future work, it would be interesting to examine any such correlation and also study the scalability of the WFST-based method for sequence alignment on longer medical event sequences and a larger dataset of patients. Further, the WFST-based method may be used to model multi-alignment tasks in other speech and language problems as well.

8 Conclusion

We propose a novel framework for aligning medical event sequences across clinical narratives based on coreference and temporal relation information using cascaded WFSTs. FSTs provide a convenient and flexible framework to model sequences of temporally ordered medical events and compose them into a combined graph representation. Decoding this graph allows us to jointly maximize coreference as well as temporal relation probabilities to derive a timeline of the most likely temporal ordering of medical events. This approach to aligning multiple sequences of medical events significantly outperforms other approaches such as dynamic programming. Moreover, we demonstrate the importance of learning temporal relations for the task timeline generation from across multiple clinical narratives by empirically proving that decoding using both coreference and temporal relation scores is far more accurate than decoding with only coreference scores.

Acknowledgments

The project was supported by Award Number Grant R01LM011116 from the National Library of Medicine. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Library of Medicine or the National Institutes of Health. The authors would like to thank Yanzhang He for his input on the WFST-based model.

[3] http://lpsolve.sourceforge.net/5.5/

References

James F. Allen. 1981. An interval-based representation of temporal knowledge. In *IJCAI*, pages 221–226.

Regina Barzilay and Kathleen R. McKeown. 2005. Sentence fusion for multidocument news summarization. *Comput. Linguist.*, 31(3):297–328, September.

Regina Barzilay, Noemie Elhadad, and Kathleen McKeown. 2002. Inferring strategies for sentence ordering in multidocument summarization. *Journal of Artificial Intelligence Research (JAIR)*, 17:35–55.

Danushka Bollegala, Naoaki Okazaki, and Mitsuru Ishizuka. 2010. A bottom-up approach to sentence ordering for multi-document summarization. *Information processing & management*, 46(1):89–109.

Philip Bramsen, Pawan Deshpande, Yoong Keok Lee, and Regina Barzilay. 2006. Inducing temporal graphs. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 189–198.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *ACL/AFNLP*, pages 602–610.

Raphael Cohen, Michael Elhadad, and Noémie Elhadad. 2013. Redundancy in electronic health record corpora: analysis, impact on text mining performance and mitigation strategies. *BMC bioinformatics*, 14(1):10.

Dina Demner-Fushman, Wendy Webber Chapman, and Clement J. McDonald. 2009. What can natural language processing do for clinical decision support? *Journal of Biomedical Informatics*, 42(5):760–772.

Quang Xuan Do, Wei Lu, and Dan Roth. 2012. Joint inference for event timeline construction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 677–687. Association for Computational Linguistics.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Association for Computational Linguistics*.

Cvetana Krstev, Duško Vitas, Ivan Obradović, and Miloš Utvić. 2011. E-dictionaries and Finite-state automata for the recognition of named entities. In *Proceedings of the 9th International Workshop on Finite State Methods and Natural Language Processing*, pages 48–56.

Shankar Kumar and William Byrne. 2003. A weighted finite state transducer implementation of the alignment template model for statistical machine translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, pages 63–70.

V Finley Lacatusu, Steven J Maiorano, and Sanda M Harabagiu. 2004. Multi-document summarization using multiple-sequence alignment. In *LREC*.

Mirella Lapata and Alex Lascarides. 2011. Learning sentence-internal temporal relations. *CoRR*, abs/1110.1394.

Mirella Lapata. 2003. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 545–552. Association for Computational Linguistics.

Zhihui Luo, Stephen B. Johnson, Albert M. Lai, and Chunhua Weng. 2011. Extracting temporal constraints from clinical research eligibility criteria using conditional random fields. In *Proc of AMIA Symposium*.

Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006. Machine learning of temporal relations. In *ACL*.

Mehryar Mohri and Fernando CN Pereira. 1998. Dynamic compilation of weighted context-free grammars. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 2*, pages 891–897. Association for Computational Linguistics.

Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. 2000. The design principles of a weighted finite-state transducer library. *Theoretical Computer Science*, 231(1):17–32.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2005. Weighted automata in text and speech processing. *CoRR*, abs/cs/0503077.

S.B. Needleman, C.D. Wunsch, et al. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453.

Cédric Notredame. 2002. Recent progress in multiple sequence alignment: a survey. *Pharmacogenomics*, 3(1):131–144.

Preethi Raghavan, Eric Fosler-Lussier, and Albert M. Lai. 2012a. Exploring semi-supervised coreference resolution of medical concepts using semantic and temporal features. In *North American Association for Computational Linguistics Annual Meeting - Human Language Technologies Conference*. Association for Computational Linguistics.

Preethi Raghavan, Eric Fosler-Lussier, and Albert M. Lai. 2012b. Inter-annotator reliability of medical events, coreferences and temporal relations in clinical narratives by annotators with varying levels of clinical expertise. In *To appear in Proceedings*

of the American Medical Informatics Association. American Medical Informatics Association.

Preethi Raghavan, Eric Fosler-Lussier, and Albert M. Lai. 2012c. Learning to temporally order medical events in clinical text. In *ACL short paper*. Association for Computational Linguistics.

Daniel Reichert, David Kaufman, Benjamin Bloxham, Herbert Chase, and Noémie Elhadad. 2010. Cognitive analysis of the summarization of longitudinal patient records. In *AMIA Annual Symposium Proceedings*, volume 2010, page 667. American Medical Informatics Association.

A. Roberts, R. Gaizauskas, M. Hepple, G. Demetriou, Y. Guo, and A. Setzer. 2008. Semantic Annotation of Clinical Text: The CLEF Corpus. In *Proceedings of the LREC 2008 Workshop on Building and Evaluating Resources for Biomedical Text Mining*, pages 19–26.

T.F. Smith and M.S. Waterman. 1981. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1).

Richard Sproat. 2006. *A Computational Theory of Writing Systems (Studies in Natural Language Processing)*. Cambridge University Press.

Marc Verhagen, Robert J. Gaizauskas, Frank Schilder, Mark Hepple, Jessica Moszkowicz, and James Pustejovsky. 2009. The tempeval challenge: identifying temporal relations in text. *Language Resources and Evaluation*, 43(2):161–179.

Lusheng Wang and Tao Jiang. 1994. On the complexity of multiple sequence alignment. *Journal of computational biology*, 1(4):337–348.

Christopher Whelan, Brian Roark, and Kemal Sonmez. 2010. Designing antimicrobial peptides with weighted finite-state transducers. In *Proceedings of IEEE Engineering in Medical Biology Society*, page 764.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Association for Computational Linguistics*, pages 189–196.

Li Zhou, Genevieve B. Melton, Simon Parsons, and George Hripcsak. 2006. A temporal constraint structure for extracting temporal information from clinical narrative. *Journal of Biomedical Informatics*, pages 424–439.

Language-Aware Truth Assessment of Fact Candidates

Ndapandula Nakashole
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA, 15213
ndapa@cs.cmu.edu

Tom M. Mitchell
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA, 15213
tom.mitchell@cs.cmu.edu

Abstract

This paper introduces FactChecker, language-aware approach to truth-finding. FactChecker differs from prior approaches in that it does not rely on iterative peer voting, instead it leverages language to infer believability of fact candidates. In particular, FactChecker makes use of linguistic features to detect if a given source objectively states facts or is speculative and opinionated. To ensure that fact candidates mentioned in similar sources have similar believability, FactChecker augments objectivity with a co-mention score to compute the overall believability score of a fact candidate. Our experiments on various datasets show that FactChecker yields higher accuracy than existing approaches.

1 Introduction

Truth-finding algorithms aim to separate true statements (facts) from false information. More specifically, given a set of statements whose truthfulness is unknown (*fact candidates*), the key goal of truth-finding algorithms is to generate a ranking such that true statements are ranked ahead of false ones. Truth-finders have the potential to address a major obstacle on the Web: the problem of sources spreading inaccurate and conflicting information. This problem continues to grow with the development of tools for easy Web authorship. Blogs, forums and social networking websites are not subject to traditional journalistic standards. Consequently, the accuracy of information reported by these sources is often unclear. Even more established newspapers and websites may sometimes report false information as they race to break stories. Therefore, truth-finding is becoming an increasingly important problem. Information extraction projects aim to distill relational facts from natural language text (Auer et al., 2007; Bollacker et al., 2008; Carlson et al., 2010; Fader et al., 2011; Nakashole et al., 2011; Del Corro and Gemulla, 2013). These projects have produced knowledge bases containing many millions of relational facts between entities. However, despite these impressive advances, there are still major limitations regarding precision. Within the context of information extraction, fact extractors assign confidence scores to extracted facts. However, such scores are often tied to the extractor's ability to *read* and *understand* natural language text. This is different from a score that indicates the degree to which a given fact candidate is *believable*. Such a believability score is sometimes also referred to as a credibility score or truthfulness score. The believability score reflects the likelihood that a given statement is true. Truth-finding algorithms aim to compute this score for each fact candidate.

Prior truth-finding methods are mostly based on iterative voting, where votes are propagated from sources to fact candidates and then back to sources (Yin et al., 2007; Galland et al., 2010; Pasternack and Roth, 2010; Li et al., 2011; Yin and Tan, 2011). At the core of iterative voting is the assumption that candidates mentioned by many sources are more likely to be true. However, additional aspects of a source influence its trustworthiness, besides external votes.

Our goal is to accurately assess truthfulness of fact candidates by taking into account the language of sources that mention them. A Mechanical Turk study we carried out revealed that there is a significant correlation between objectivity of language and trustworthiness of sources. Objectivity of language refers to the use of neutral, impartial language, which is not personal, judgmental, or emotional. Trustworthiness refers to

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1009–1019,
Baltimore, Maryland, USA, June 23-25 2014.

a source of information being reliable and truthful. We use linguistics features to detect if a given source objectively states facts or is speculative and opinionated. Additionally, in order to ensure that fact candidates mentioned in similar sources have similar believability scores, our believability computation model incorporates influence of co-mentions. However, we must avoid falsely boosting co-mentioned fact candidates. Our model addresses potential false boosts in two ways: first, by ensuring that co-mention influence is only propagated to related fact candidates; second, by ensuring that the degree of co-mention influence is determined by the trustworthiness of the sources in which co-mentions occur.

The contribution of this paper is a language-aware truth-finding approach. More precisely, we make the following contributions: (1) *Alternative Fact Candidates:* Truth-finders rank a given fact candidate with respect to its alternatives. For example, alternative places where Barack Obama could have been born. Virtually all existing truth-finders assume that the alternatives are provided. In contrast, we developed a method for generating alternative fact candidates. (2) *Objectivity-Trustworthiness Correlation:* We hypothesize that objectivity of language and trustworthiness of sources are positively correlated. To test this hypothesis, we designed a Mechanical Turk study. The study showed that this correlation does in fact hold. (3) *Objectivity Classifier:* Using labeled data from the Mechanical Turk study, we developed and trained an objectivity classifier which performed better than prior proposed lexicons from literature. (4) *Believability Computation:* We developed FactChecker, a truth-finding method that linearly combines objectivity and co-mention influence. Our experiments showed that FactChecker outperforms prior methods.

2 Fact Candidates

In this section, we formally define what constitutes a fact candidate and describe how we go about understanding semantics of fact candidates. We then present our approach for generating alternative fact candidates.

2.1 Representation

The *triple* format is the most common representation of facts in knowledge bases. A formal specification of the triple format is presented in the RDF primer[1]. In RDF, data is represented as subject-predicate-object (SPO) triples. In this work, we restrict predicates to *verbs* (or *verbal phrases* such as "plays for", "graduated from", etc.). Literature on automatic relation discovery (Fader et al., 2011) has shown that verbal phrases uncover a large fraction of binary predicates while reducing the amount of noisy phrases that do not denote any relations. Therefore, we define a fact candidate as follows:

Definition 1 (Fact Candidate) *A fact candidate f_i is an $\langle S \rangle$ V $\langle O \rangle$ triple; where S is the subject, V is a verbal phrase, and O is the object. We aim to compute the truthfulness of f_i, $\tau(f_i) \in \{T, F\}$, where T and F stand for $true$ and $false$, respectively.*

Note that in this paper we are interested in cases where $\tau(f_i)$ is either T *or* F. That is, we assess truthfulness of *factual* statements and not opinions whose truthfulness is often both T and F to some degree. For example, the triples: $\langle$*Obama*$\rangle$ *born in* $\langle$*Kenya*$\rangle$ and $\langle$*Obama*$\rangle$ *graduated from* $\langle$*Harvard*$\rangle$ are valid fact candidates. However, the triple: $\langle$*Obama*$\rangle$ *deserves* $\langle$*Nobel Peace Prize*$\rangle$ is not.

2.2 Semantics

Based on the SVO triple, the meaning of a fact candidate can be unclear and ambiguous. Therefore, we first determine the semantics of a fact candidate before computing its truthfulness.

Entity Types. We first determine the expected types of the subject and object in the SVO. For example, for the SVO $\langle$*Einstein*$\rangle$ *died in* $\langle$*Princeton*$\rangle$, the expected types are *person* $\times$ *location*. We determine this by first computing the types of entities that are valid for each verb (verbal phrase) in a large SVO collection of 114m SVO triples (Talukdar et al., 2012). Typing verbal phrases is a once-off computation. Our phrase typing method is similar to prior work on typing relational phrases (Nakashole et al., 2012). Examples of typed phrases are: $\langle$*person*$\rangle$ *died in* $\langle$*year*$\rangle$, $\langle$*person*$\rangle$ *died in* $\langle$*location*$\rangle$, and $\langle$*athlete*$\rangle$ *plays for* $\langle$*team*$\rangle$. Given a triple, we look up the types for the subject and the object and then determine which of the typed phrases are compatible with the current triple. We look up entity types in a knowledge

[1] http://www.w3.org/TR/rdf-primer/

base containing entities and their types. In particular, we use the NELL entity typing API (Carlson et al., 2010). NELL's entity typing method has high recall because when entities are not in the knowledge base, it performs on-the-fly type inference using the Web. This is not the case for other options such as (Auer et al., 2007; Bollacker et al., 2008; Hoffart et al., 2011).

Relation Cardinality. Next, we learn cardinalities of verbal phrases. Cardinality refers to how arguments of a given relation relate to one another numerically. We define the relation cardinality of a verb $Card(V)$, as the average number of expected arguments per given subject. For example, for the relation "died in", 1 location is expected for each subject. For other relations, the expected number of arguments can be greater than 1 but less than $n : n \in \mathbb{R}, n > 1$. We approximate n using statistics from the 114m SVO corpus based on the average number of arguments per given first argument. In a once-off computation, we generate cardinality approximations per typed verbal phrase V and its inverse V^{-1}. For example, we generate the cardinality estimates for both: ⟨*person*⟩ *died in* ⟨*location*⟩ and for ⟨*location*⟩ *INVERSE-OF(died in)* ⟨*person*⟩.

Synonymous Relations. Natural language is diverse. Semantically similar phrases can be syntactically different. Therefore, we learn other verbs that can be used to substitute V in SVO. We pre-compute synonymous phrases from the 114m SVO corpus using distributional semantics in the same spirit as (Lin and Pantel, 2001; Nakashole et al., 2012).

Synonymous verbs, relation cardinalities, and entity types enable us to generate alternative fact candidates.

2.3 Alternative Fact Candidates

Truth-finding methods rank f_i relative to alternative candidates. While prior methods assume the alternatives are known apriori, we developed a method for generating alternative fact candidates. For a given f_i, we first identify the *fixed argument*. The fixed argument is the argument of the SVO which when fixed, requires finding the fewest number of alternative candidates. For example, for ⟨*Einstein*⟩ *died in* ⟨*Princeton*⟩, the solution is to fix the subject. This is because the cardinality of ⟨*person*⟩ *died in* ⟨*location*⟩ is one (1). On the other hand, the cardinality of "INVERSE-OF(died in)" is many(n). In other words, the number of places where a person can be born (one) is much fewer than the number of people that can die in a place (many). In our example, alternatives are possible places, other than Princeton, where Einstein could have died. For example: ⟨*Einstein*⟩ *died in* ⟨*Germany*⟩ or ⟨*Einstein*⟩ *died in* ⟨*Switzerland*⟩. More generally, the fixed argument of fact candidate f_i, is defined as follows:

Definition 2 (Fixed Argument) *Let Card(V) be the cardinality of* V *and* $Card(V^{-1})$ *be the cardinality of the inverse of* V*, if* $Card(V) < Card(V^{-1})$*, then the fixed argument is the subject,* $Arg^{fixed}(f_i) = S$*, else it is the object,* O*. If* $Card(V) == Card(V^{-1})$*, then both arguments are fixed, one at a time.*

We use the fixed argument to define a *topic* as the fixed argument plus the verb. Therefore, for the SVO ⟨*X*⟩ *died in* ⟨*Y*⟩, the topic "places where X died", ($Arg^{fixed} = S$), is not the same as the topic "people who died in Y" ($Arg^{fixed} = O$).

To locate alternatives, we use the topic ($Arg^{fixed} + V$) as a query. We search three sources to either locate relevant documents or relevant triples: the Google Web search API, the 114m SVO collection, and the NELL KB. The SVO collection and the KB return triples, however, the Web search API returns documents. Therefore, we apply a triple extractor to the retrieved documents. For all potential alternative triples, we perform type checking to ensure that the arguments of the triples are type-compatible with f_i. Furthermore, we generate an additional query for every synonymous verb sV_i, replacing V with sV_i. Example queries are: "Einstein died in", "Einstein passed in", etc.

3 Objectivity and Trustworthiness

The principle of *objective journalism*, which is a significant part of journalistic ethics, aims to promote factual and fair reporting, undistorted by emotion or personal bias (Schudson, 1978; Kaplan, 2002). Objectivity is also required in reference sources such as encyclopedias, scientific publications, and textbooks. For example, Wikipedia enforces a neutral point-of-view policy (NPOV)[2]. Articles violating the NPOV policy are marked

[2]http://en.wikipedia.org/wiki/Wikipedia:Neutral_point_of_view

to indicate potential bias. While opinions, emotions, and speculations can also be expressed using objective language, they are often stated using subjective language (Turney et al., 2002; Riloff and Wiebe, 2003; Yu and Hatzivassiloglou, 2003; Wiebe et al., 2004; Liu et al., 2005; Recasens et al., 2013). For example, consider the following pieces of text:

(S) Well, I think Obama was born in Kenya because his grandma who lives in Kenya said he was born there.

(O) Theories allege that Obama's published birth certificate is a forgery, that his actual birthplace is not Hawaii but Kenya.

Text S is a snippet from Yahoo Answers and text O is a snippet from the Wikipedia page titled: "Barack Obama Citizenship Conspiracy Theories". S is subjective, expressing the opinion of the author. On the other hand, O is objective, stating only what has been alleged. Literature on sentiment analysis (Turney et al., 2002; Liu et al., 2005), subjectivity detection (Riloff and Wiebe, 2003; Wiebe et al., 2004), and bias detection (Yu and Hatzivassiloglou, 2003; Recasens et al., 2013) has developed lexicons for identifying subjective language. Due to the principle of objective journalism and the requirement of objectivity placed on reference sources, we hypothesize a link between objectivity and trustworthiness as follows.

Hypothesis 1 *Objective sources are more trustworthy than subjective sources. Therefore, we can assume that fact candidates stated in objective sources are more likely to be true than those stated in subjective sources.*

To test the validity of the hypothesis, we carried out a study where we solicited human input.

3.1 Mechanical Turk Study

We deployed an annotation study on Amazon Mechanical Turk (MTurk)[3], a crowd-sourcing platform for tasks requiring human input. Tasks on MTurk are small questionnaires consisting of a description and a set of questions. Our study consisted of two independent tasks. The first task was titled "Trustworthiness of News Articles", where annotators were given a link to a news article and asked to judge if they thought it was trustworthy or not. The second task was titled "Objectivity of News Articles". For this task, annotators were asked to judge if a given article is objective or subjective. For both tasks a third option of "not sure" was provided. We randomly selected 500 news articles from a corpus of about 300,000 news articles obtained from Google News from the topics of Top News, Business, Entertainment, and SciTech. For each task, every article was judged by three annotators. This produced a total of 3000 annotations. When we analyzed the output, we accepted a label as valid for a given article if the label was selected by the majority of the judges. Based on this criteria, we obtained a set of 420 articles that were both labeled for trustworthiness and objectivity.

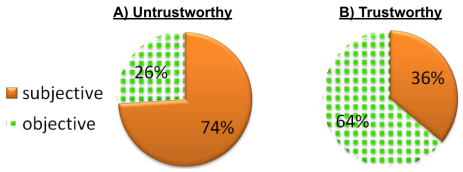


Figure 1: Summary of the results of the annotation study on objectivity and trustworthiness.

A summary of the outcome of the study is shown in Figure 1; 74% of the untrustworthy articles were independently labeled as subjective. On the other hand, 64% of trustworthy articles were independently labeled as objective. These results indicate a non-trivial positive correlation between objectivity and trustworthiness. We leverage this correlation in our believability computation model. To incorporate objectivity in FactChecker, we require for a given source document, an objectivity score $\in [0, 1]$, where 0 means the source is subjective and 1 means it is objective. Next, describe our method for automatically determining objectivity of sources.

3.2 Automatic Objectivity Detection

We trained a logistic regression classifier to predict the objectivity of a document. For training and testing data, we used the labeled data from the Mechanical Turk study. We additionally used labeled text from prior work on subjectivity detection (Pang and Lee, 2004). This resulted in a total of $4,600$ documents, half subjective and the other half objective. We used 4000 documents for

[3]http://www.mturk.com

#	Feature
1	Subjectivity lexicon of strong and weak subjective words (Riloff and Wiebe, 2003).
2	Sentiment lexicon of positive and negative words (Liu et al., 2005).
3	Wikipedia-derived bias lexicon (Recasens et al., 2013).
4	Part-of-speech (POS) tags
5	Frequent bi-grams

Table 1: Features used for the objectivity detector.

training, 2000 per label. The rest of the documents were split into a development set (380) and a test set (220).

A summary of the features we used is shown in Table 1. Features 1-3 refer to lexicons developed by prior methods on subjectivity (Wiebe et al., 2004), sentiment analysis (Liu et al., 2005) and bias detection (Recasens et al., 2013). Feature 4 refers to part-of-speech tags of the terms found in the document that are also in the lexicons. Feature 5 refers to bi-grams that frequently occur (mention frequency of > 10) in the $4,600$ documents. The most contributing features were the lexicons, features (1-3) and the frequent bi-grams, feature 5. We discovered that using *frequent* bi-gram features instead of uni-grams or bi-grams resulted in higher precision. The classifier was able to determine that for example bi-grams such as "think that", "so funny" and "you thought" are negative features for objectivity. Evaluation results of our objectivity detector vs. baselines are shown in Table 2. FactChecker's objectivity detector has precision of 0.7814 ± 0.0539, with a 0.9-confidence Wilson score interval (Brown et al., 2001) and this outperforms the baselines. Next, we describe how we leverage objectivity into FactChecker's truthfulness model.

4 Believability Computation Model

FactChecker computes the believability score of a fact candidate from its: i) objectivity score and (ii) co-mention score. In this section we define each of these scores.

The objectivity score reflects the trustworthiness of sources where a fact candidate is mentioned. Given a fact candidate f_i, mentioned in a set of documents D_i, where each document $d \in D_i$ has objectivity $\mathcal{O}(d)$, f_i's objectivity score is defined as follows:

Approach	Accuracy
Sentiment Lexicon	0.65±0.06
Wikipedia bias Lexicon	0.69±0.06
Subjectivity Lexicon	0.70±0.06
FC-Objectivity Detector	**0.78**±0.05

Table 2: Accuracy of the objectivity detector.

Definition 3 (Objectivity Score)

$$\mathcal{O}(f_i) = log|D_i|.\frac{\sum_{d_k \in D_i} \mathcal{O}(d_k)}{|D_i|} \quad (1)$$

We do not use the sum of objectivity of sources as the objectivity score because this enables fact candidates mentioned in many low objectivity sources to have high aggregate objectivity. Similarly, we avoid using average objectivity of the sources as it overestimates objectivity of candidates stated in few sources. A candidate mentioned in 10 sources with 0.9 objectivity should have higher objectivity than a candidate stated in 1 source of 0.9 objectivity. In Equation 1, $log|D_i|$ addresses this issue.

The co-mention score aims to ensure that fact candidates mentioned in similar sources have similar believability scores. Suppose candidate f_i is mentioned in many highly objective sources, another candidate f_j is stated in only one highly objective source d_k where f_i is also mentioned. Then the believability of f_j should be boosted by it being co-mentioned with f_i. If on the other hand f_i and f_j were co-mentioned in a subjective source, f_j should receive less boost from f_i. This leads us to the co-mention score $\mu(f_i)$ of a candidate.

Definition 4 (Co-Mention Score)

$$\mu(f_i) = \rho(f_i) + \sum_{f_j \in \mathcal{F}} w_{ij}\mu(f_j) \quad (2)$$

Where $\rho(f_i)$ is the normalized mention frequency of f_i. The propagation weight w_{ij} controls how much boost is obtained from a co-mentioned candidate. We define propagation weight, w_{ij}, as the average of the objectivity of the sources that mention both candidates.

$$w_{ij} = average\ \mathcal{O}(d_k) : d_k \in (D_i \cap D_j) \quad (3)$$

where $\mathcal{O}(d_k)$ is the objectivity of document d_k, D_i and D_j are the sets of documents that mention f_i and f_j, respectively. Notice that we could boost co-mentioned but not related candidates, thereby causing false boosts. To remedy this, we only allow w_{ij} to be greater than $zero$ if the fact candidates f_i and f_j are on the same topic. Recall that the topic is determined by the fixed argument (Definition 2) and the verb. Allowing only fact candidates on the same topic to influence each other is important considering that many trivial facts are often repeated in sources of diverse quality.

To leverage the inter-dependencies among *related* co-mentioned fact candidates, we model the solution with a graph ranking method. Each fact candidate is a node and there is an edge between each pair of related fact candidate nodes f_i and f_j, with w_{ij} as the edge weight. Thus, equation 2 can be reformulated as $\mu = M\mu$, where μ is the co-mention score vector and M is a Markov matrix which is stochastic, irreducible and aperiodic. Thus, a power method will converge to a solution in a similar manner to PageRank. Implementation consists of iteratively applying Equation 2 until the change in the score is less than a threshold ϵ. The solution is the final co-mention scores of fact candidates.

Finally, to compute the believability score of a fact candidate, we linearly combine its objectivity score with its co-mention as follows:

Definition 5 (Believability Score)

$$\beta(f_i) = \lambda\mathcal{O}(f_i) + (1-\lambda)\mu(f_i) \qquad (4)$$

Where λ is a weighting parameter $\in [0,1]$ which controls the relative importance of the two aspects of FactChecker. As we show in our experiments, λ can be robustly chosen within the range of 0.2 to 0.6. In our experiments we used $\lambda = 0.6$.

The entire procedure of FactChecker is summarized in Algorithm 1.

Algorithm 1 FactChecker

Input: A set $\mathcal{F}$ of fact candidates
Input: KB $\mathcal{K}$, SVO corpus $\mathcal{C}$, Web $\mathcal{W}$
Output: A set $\mathcal{L}$ of rankings $\forall f_i \in F$
$\mathcal{L} = \emptyset$
while $\mathcal{F} \neq \emptyset$ **do**
  pick f_i from $\mathcal{F}$
  $\mathcal{A}$= *getAlternatives*(f_i,$\mathcal{K}$,$\mathcal{C}$,$\mathcal{W}$)
  $PriorityQueue\ L_i = \emptyset$
  for all alternative fact candidates $f'_j \in \mathcal{A}$ **do**
    $\beta(f'_j)$ = *getBelievabilityScore*(f'_j)
    $L_i.insert(f'_j, \beta(f'_j))$
  end for
  $\beta(f_i)$ = *getBelievabilityScore*(f_i)
  $L_i.insert(f_i, \beta(f_i))$
  $\mathcal{L} \cup L_i$
  Remove f_i from F
end while
return $\mathcal{L}$

5 Evaluation

We evaluated FactChecker for accuracy. We define accuracy as the probability of a true fact candidate having a higher believability score than a false candidate. Let $\tau(f_i) \in \{T, F\}$ be the truthfulness of a fact candidate f_i, accuracy is defined as:

$$Acc = \frac{\sum_{(\tau(f_i)=T:\tau(f_j)=F)}(\beta(f_i) > \beta(f_j))}{|\{\forall(f_i,f_j) : \tau(f_i) = T \wedge \tau(f_j) = F\}|}$$

Datasets. We evaluated FactChecker on three datasets: **i) KB Fact Candidates:** The first dataset consists of fact candidates taken from the fact extraction pipeline of a state-of-the-art knowledge base, NELL (Carlson et al., 2010). The fact candidates span four different relation types: company acquisitions, book authors, movie directors and athlete teams. For each fact candidate, we applied our alternative candidate generation method. We only considered fact candidates with non-trivial alternative candidate sets; where the alternative candidate set is greater than *zero*. Since all of the baselines we compared against assume alternatives are provided, we apply all methods to the same set of alternative fact candidates discovered by our method. Details of this dataset are shown as rows starting with "KB-" in Table 3.

ii) Wikipedia Fact Candidates: For the second dataset, we did not restrict the fact candidates to specific topics from a knowledge base, instead we aimed to evaluate all fact candidates about a given entity. We selected entities from Wikipedia. For this, we chose US politicians: all current state senators, all current state governors, and all 44 presidents. First, we extracted fact candidates

	#Candidates	#Alternatives
KB-Acquisitions	50	241
KB-Authors	50	295
KB-Directors	50	228
KB-Teams	40	162
WKP Politicians	54	219
GK Quiz	18	72

Table 3: Fact candidate datasets.

from the infoboxes of the Wikipedia pages of the entities. Second, we applied our alternative candidate generation method to discover alternatives from the Web, SVO corpus, and NELL. Details of the resulting dataset are shown in the row "WKP Politicians" in Table 3.

iii) General Knowledge Quiz: The third dataset consists of questions from a general knowledge quiz [4]. We selected questions from the inventions category. Questions are multiple choice, with 4 options per question. Thus, from each question, we created one fact candidate and 3 alternative candidates. Details of the resulting dataset are shown in the row "KWP Quiz" in Table 3.

Baselines. We compared FactChecker against five baselines: **i) Vote** counts the number of sources that mention the fact candidate. **ii) TruthFinder** is an iterative voting approach where votes are propagated from sources to fact candidates and then back to sources. Implemented as described in (Yin et al., 2007). **iii) Investment** is also based on transitive voting, however scores are updated differently. A source gets a vote of trust from each candidate it "invests" in, but the vote is weighted by the proportion of trust the source previously "invested" in the candidate relative to other investors. Implemented as described in (Pasternack and Roth, 2010). **iv) PooledInvest** is a variation of investment, we report both because in their paper, there was no clear winner among the two variations. **v) 2-Estimates** is a probabilistic model which approximates error rates of sources and fact candidates (Galland et al., 2010).

5.1 Accuracy on KB Fact Candidates

Figure 2 shows accuracy on KB fact candidates. FactChecker achieves accuracy between 70% and 88% and is significantly more accurate than the other approaches on all relations except company acquisitions. On book authors, movie directors, and athlete teams, FactChecker outperforms all other approaches by at least 10%, 9%, and 8% respectively. On company acquisitions, the different methods achieve similar accuracy, with TruthFinder being the most accurate and FactChecker is 4% behind. Company acquisitions also yield the lowest difference between Vote and the highest performing method, of 6%. For book authors, movie directors, and athlete teams, the difference between majority Vote and the highest performing method (FactChecker in this case) is 13%, 12%, and 13% respectively.

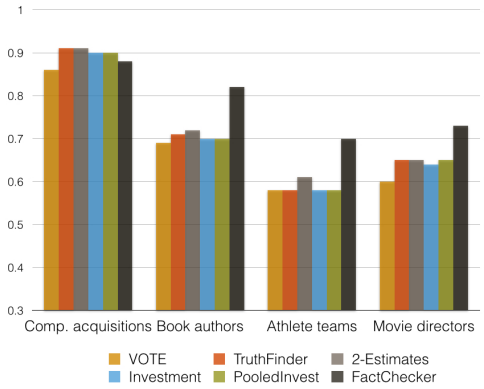


Figure 2: Accuracy of KB fact candidates.

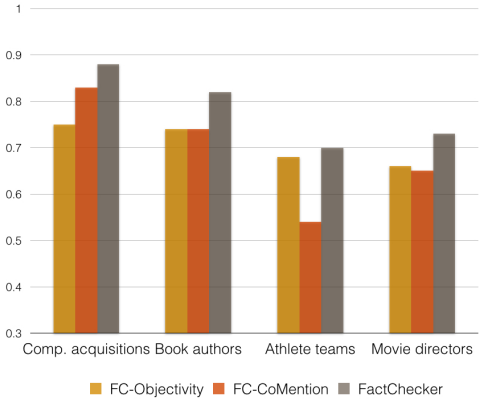


Figure 3: FactChecker variations.

5.2 Accuracy of FactChecker Variations

To quantify how various aspects of our approach affect overall performance, we studied two variations. The first variation is *FC-Objectivity* which only uses objectivity to compute believability. Thus, $\lambda = 1$ in Definition 5. The second variation is *FC-CoMention* which only uses co-mention scores to compute believability, $\lambda = 0$. The

[4] http://www.indiabix.com/general-knowledge/

Approach	WKP Politicians	GK Quiz
Vote	0.85±0.09	0.82±0.15
TruthFinder	0.85±0.09	0.82±0.15
2-Estimates	0.85±0.09	0.82±0.15
Investment	0.86±0.08	0.82±0.15
PooledInvest	0.85±0.09	0.82±0.15
FC-Objectivity	0.88±0.08	**0.87**±0.12
FC-CoMention	0.85±0.09	0.72±0.18
FactChecker	**0.90**±0.07	**0.87**±0.12

Table 4: Accuracy on politicians and quiz data sets

last variation is the full FactChecker method using both objectivity and co-mentions with $\lambda = 0.6$ From Figure 3, it is clear that both the objectivity of sources and the influence of co-mentions contribute to the overall accuracy of FactChecker. Full-fledged FactChecker performs better than both variations. In most cases, *FC-Objectivity* performs better than *FC-CoMention*.

5.3 Accuracy on Wikipedia Fact Candidates

Table 4, column "WKP Politicians", shows accuracy on Wikipedia fact candidates, with a 0.9-confidence Wilson score interval (Brown et al., 2001). For this dataset we again see FactChecker outperforming the other methods under comparison. On this dataset, FactChecker has a accuracy of 0.9 ± 0.07 and a 5% accuracy advantage over the other methods. The second best performance comes from the *FC-Objectivity* variation, with accuracy of 0.88 ± 0.08.

5.4 Accuracy on General Knowledge Quiz

Table 4, column "GK Quiz ", shows accuracy on the general knowledge quiz fact candidates. On this dataset, FactChecker and its objectivity-only variation (FC-objectivity) have the highest accuracy of 87%. Notice that this dataset was the only one where we did not generate the alternative fact candidates. Instead, we took the options of the multiple choice questions as alternatives. Since the quiz is meant to be taken by humans, the alternatives are often very close, plausible answers. Yet even in this difficult setting, we see FactChecker outperforming the baselines.

Sample fact candidates, with ranked alternatives from all three datasets are shown in Table 5.

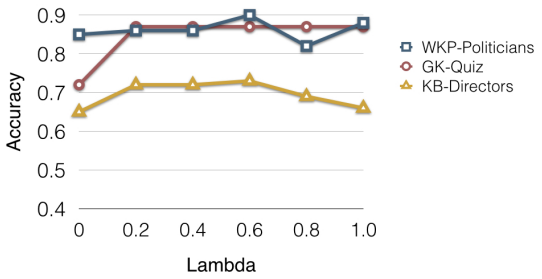


Figure 4: Effect of λ of FactChecker.

5.5 Parameter Sensitivity

We analyzed the effect of the selection of lambda λ (see Definition 5) on FactChecker's performance. The result of this analysis is shown in Figure 4. FactChecker is insensitive to this parameter when λ is varied from 0.2 to 0.6. Therefore, lambda can be robustly chosen within this range.

5.6 Discussion

Overall, from these results we make the following observations: i) Majority vote is a competitive baseline; ii) Iterative voting-based methods provide slight improvements on majority vote. This is due to the fact that at the core of iterative voting is still the assumption that fact candidates mentioned in many sources are more likely to be true. Therefore, for both majority vote and iterative voting, when mention frequencies of various alternatives are the same, accuracy suffers. Based on these observations, it is clear that truth-finding solutions need to incorporate fine-grained content-aware features outside of external votes. FactChecker takes a step in this direction by incorporating the document-level feature of objectivity.

6 Related Work

There is a fairly small body of work on truth-finding (Yin et al., 2007; Galland et al., 2010; Pasternack and Roth, 2010; Li et al., 2011; Yin and Tan, 2011; Zhao et al., 2012; Pasternack and Roth, 2013). The method underlying most truth-finding algorithms is iterative transitive voting (Yin et al., 2007; Galland et al., 2010; Pasternack and Roth, 2010; Li et al., 2011). Fact candidates are initialized with a score. Trustworthiness of sources is then computed from the believability of the fact candidates they mention. In return, believability of candidates is recomputed based on the trustworthi-

Dataset	Fact Candidate	Alternatives & Ranking
WKP	⟨*George W. Bush*⟩ *lived in* ⟨*Midland,TX*⟩	1.Midland,TX 2.Compton,CA 3.Washington D.C. 4.Venezuela*
KB	⟨*Dirk Kuyt*⟩ *plays for* ⟨*Liverpool*⟩	1. Liverpool 2.Cardiff City* 3.Netherlands 4.Hungary*
Quiz	⟨*Bifocals*⟩ *invented by* ⟨*Benjamin Franklin*⟩	1. Benjamin Franklin 2. Rudolf Diesel* 3.Thomas Alva Edison* 4.Alfred B. Nobel*

Table 5: Sample rankings by FactChecker, alternatives marked (*) are false. The ranking of the candidate from the "KB" dataset is not completely accurate.

ness of their sources. This process is repeated over several iterations until convergence. (Yin et al., 2007) was the first to implement this idea, subsequent work improved upon iterative voting in several directions. (Dong et al., 2009) incorporates copying-detection; giving high trust to sources that are independently authored. (Galland et al., 2010) approximates error rates of sources and fact candidates. (Pasternack and Roth, 2010) introduces prior knowledge in the form of linear programming constraints in order to ensure that the truth discovered is consistent with what is already known. (Yin and Tan, 2011) introduces supervision by using ground truth facts so that sources that disagree with the ground truth are penalized. (Li et al., 2011) uses search engine APIs to gather additional evidence for believability of fact candidates. WikiTrust (Adler and Alfaro, 2007) is a content-aware but domain-specific method. It computes trustworthiness of wiki authors based on the revision history of the articles they have authored. Motivated by interpretability of probabilistic scores, two recent papers addressed the truth-finding problem as a probabilistic inference problem over the sources and the fact candidates (Zhao et al., 2012; Pasternack and Roth, 2013). Truth-finders based on textual entailment such as TruthTeller (Lotan et al., 2013) determine if a sentence states something or not. The focus is on understanding natural language, including the use of negation. This is similar to the goal of fact extraction (Banko et al., 2007; Carlson et al., 2010; Fader et al., 2011; Nakashole et al., 2011; Del Corro and Gemulla, 2013).

In a departure from prior work, our method leverages language of sources in its believability computation model. Furthermore, we introduced a co-mention score which is designed to avoid potential false boots among fact candidates. Additionally, we developed a method for generating alternative fact candidates. Prior methods assume these are readily available. Only (Li et al., 2011) uses the Web to identify alternatives, however, this is only done after manually specifying the fixed argument. In contrast, we introduced a method for identifying the fixed argument based on relation cardinalities learned from SVO statistics.

7 Conclusion

In this paper, we presented FactChecker, a language-aware approach to truth-finding. In contrast to prior approaches, which rely on external votes, FactChecker includes objectivity of sources in its believability computation model.

FactChecker can be seen as a first step towards language-aware truth-finding. Future directions include using more sentence-level features such the use of hedges, assertive verbs, and factive verbs. These types of words fall into a class of words used to express certainties, speculations or doubts — these are important cues that FactChecker can leverage.

Acknowledgments

We thank members of the NELL team at CMU for their helpful comments. This research was supported by DARPA under contract number FA8750-13-2-0005.

References

S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z.G. Ives: DBpedia: A Nucleus for a Web of Open Data. In *Proceedings of the 6th International Semantic Web Conference (ISWC)*, pages 722–735, Busan, Korea, 2007.

B. T. Adler, L. de Alfaro: A content-driven reputation system for the wikipedia. In *Proceedings of the 16th International Conference on World Wide Web (WWW)*, pages 261-270, 2007.

M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, O. Etzioni: Open Information Extraction from the Web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2670–2676, Hyderabad, India, 2007.

K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor: Freebase: a Collaboratively Created Graph Database for Structuring Human Knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages, 1247-1250, Vancouver, BC, Canada, 2008.

L. D. Brown, T.T. Cai, A. Dasgupta: Interval Estimation for a Binomial Proportion. Statistical Science 16: pages 101–133, 2001.

E. Cabrio, S. Villata: Combining Textual Entailment and Argumentation Theory for Supporting Online Debates Interaction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 208-212, 2012.

A. Carlson, J. Betteridge, R.C. Wang, E.R. Hruschka, T.M. Mitchell: Coupled Semi-supervised Learning for Information Extraction. In *Proceedings of the Third International Conference on Web Search and Web Data Mining (WSDM)*, pages 101–110, New York, NY, USA, 2010.

L. Del Corro, R. Gemulla: ClausIE: clause-based open information extraction. In *Proceedings of the 22nd International Conference on World Wide Web (WWW)*, pages 355-366. 2013.

X. Dong, L. Berti-Equille, D. Srivastava: Truth discovery and copying detection in a dynamic world. In *Proceedings of the VLDB Endowment* PVLDB, 2(1), pp. 562-573, 2009.

A. Fader, S. Soderland, O. Etzioni: Identifying Relations for Open Information Extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1535–1545, Edinburgh, UK, 2011.

A. Galland, S. Abiteboul, A. Marian, P. Senellart: Corroborating information from disagreeing views. In *Proceedings of the 3rd International Conference on Web Search and Web Data Mining (WSDM)*, pages 131-140, 2010.

C. Havasi, R. Speer, J. Alonso: ConceptNet 3: a Flexible, Multilingual Semantic Network for Common Sense Knowledge. In *Proceedings of the Recent Advances in Natural Language Processing (RANLP)*, Borovets, Bulgaria, 2007.

J. Hoffart, F. Suchanek, K. Berberich, E. Lewis-Kelham, G. de Melo, G. Weikum: YAGO2: Exploring and Querying World Knowledge in Time, Space, Context, and Many Languages. In *Proceedings of the 20th International Conference on World Wide Web (WWW)*, pages 229–232, Hyderabad, India. 2011.

R. Kaplan: Politics and the American Press: The Rise of Objectivity, pages 1865-1920, New York, Cambridge University Press, 2002.

X. Li and W. Meng, C. T. Yu: T-verifier: Verifying truthfulness of fact statements. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pp. 63-74, 2011.

D. Lin, P. Pantel: DIRT: discovery of inference rules from text. KDD 2001

B. Liu, M. Hu, J. Cheng: Opinion Observer: analyzing and comparing opinions on the Web. In*Proceedings of the 14th International Conference on World Wide Web (WWW)*, pages 342351, 2005.

A. Lotan, A. Stern, I. Dagan TruthTeller: Annotating Predicate Truth. In *Proceedings of Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*, pp. 752-757, 2013.

N. Nakashole, M. Theobald, G. Weikum: Scalable Knowledge Harvesting with High Precision and High Recall. In *Proceedings of the 4th International Conference on Web Search and Web Data Mining (WSDM)*, pages 227–326, Hong Kong, China, 2011.

N. Nakashole, T. Tylenda, G. Weikum: Fine-grained Semantic Typing of Emerging Entities. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1488-1497, 2013.

N. Nakashole, G. Weikum, F. Suchanek: PATTY: A Taxonomy of Relational Patterns with Semantic Types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1135 - 1145, Jeju, South Korea, 2012.

V. Nastase, M. Strube, B. Boerschinger, C. Zirn, A. Elghafari: WikiNet: A Very Large Scale Multi-Lingual Concept Network. In *Proceedings of the 7th International Conference on Language Resources and Evaluation(LREC)*, Malta, 2010.

B. Pang, L. Lee: A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based

on Minimum Cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 271-278, 2004.

J. Pasternack, D. Roth: Knowing What to Believe. In *Proceedings the International Conference on Computational Linguistics (COLING)*, pp. 877-885, Beijing, China. 2010.

J. Pasternack, D. Roth: Latent credibility analysis. In *Proceedings of the 22nd International Conference on World Wide Web (WWW)*, pp. 1009-1020, 2013.

E. Riloff, J. Wiebe: Learning Learning extraction patterns for subjective expressions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 105112, 2013.

M. Recasens, C. Danescu-Niculescu-Mizil, D. Jurafsky: Linguistic Models for Analyzing and Detecting Biased Language. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1650-1659, 2013.

F. Niu, C. Zhang, C. Re, J. W. Shavlik: DeepDive: Web-scale Knowledge-base Construction using Statistical Learning and Inference. In the VLDS Workshop, pages 25-28, 2012.

M. Schudson: Discovering the News: A Social History of American Newspapers. New York: Basic Books. 1978.

F. M. Suchanek, M. Sozio, G. Weikum: SOFIE: A Self-organizing Framework for Information Extraction. In*Proceedings of the 18th International Conference on World Wide Web (WWW)*, pages 631–640, Madrid, Spain, 2009.

P. P. Talukdar, D. T. Wijaya, T.M. Mitchell: Acquiring temporal constraints between relations. In *Proceeding of the 21st ACM International Conference on Information and Knowledge Management*, pages 992-1001, CIKM 2012.

P. D. Turney: Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 417424. 2002.

J. Wiebe, T. Wilson, R. Bruce, M. Bell, M. Martin: Learning subjective language. Computational Linguistics, 30(3):277308. 2004.

X. Yin, J. Han, P. S. Yu: Truth Discovery with Multiple Conflicting Information Providers on the Web. In *Proceedings of the International Conference on Knowledge Discovery in Databases (KDD)* , pages1048-1052. 2007.

X. Yin, W. Tan: Semi-supervised truth discover. In *Proceedings of the 19th International Conference on World Wide Web (WWW)*, pp. 217-226, 2011.

H. Yu, V. Hatzivassiloglou: Towards Answering Opinion Questions: Separating Facts from Opinions and Identifying the Polarity of Opinion Sentences. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages. 129-136, 2003.

B. Zhao, B. I. P. Rubinstein, J. Gemmell, J. Han: A Bayesian approach to discovering truth from conflicting sources for data integration. In *Proceedings of the VLDB Endowment (PVLDB)*, 5(6):550-561, 2012.

That's sick dude!: Automatic identification of word sense change across different timescales

Sunny Mitra[1], **Ritwik Mitra**[1], **Martin Riedl**[2],
Chris Biemann[2], **Animesh Mukherjee**[1], **Pawan Goyal**[1]
[1]Dept. of Computer Science and Engineering,
Indian Institute of Technology Kharagpur, India – 721302
[2] FG Language Technology, Computer Science Department, TU Darmstadt, Germany
[1]{sunnym,ritwikm,animeshm,pawang}@cse.iitkgp.ernet.in
[2]{riedl,biem}@cs.tu-darmstadt.de

Abstract

In this paper, we propose an unsupervised method to identify noun sense changes based on rigorous analysis of time-varying text data available in the form of millions of digitized books. We construct distributional thesauri based networks from data at different time points and cluster each of them separately to obtain word-centric sense clusters corresponding to the different time points. Subsequently, we compare these sense clusters of two different time points to find if (i) there is birth of a new sense or (ii) if an older sense has got split into more than one sense or (iii) if a newer sense has been formed from the joining of older senses or (iv) if a particular sense has died. We conduct a thorough evaluation of the proposed methodology both manually as well as through comparison with WordNet. Manual evaluation indicates that the algorithm could correctly identify 60.4% birth cases from a set of 48 randomly picked samples and 57% split/join cases from a set of 21 randomly picked samples. Remarkably, in 44% cases the birth of a novel sense is attested by WordNet, while in 46% cases and 43% cases split and join are respectively confirmed by WordNet. Our approach can be applied for lexicography, as well as for applications like word sense disambiguation or semantic search.

1 Introduction

Two of the fundamental components of a natural language communication are word sense discovery (Jones, 1986) and word sense disambiguation (Ide and Veronis, 1998). While discovery corresponds to acquisition of vocabulary, disambiguation forms the basis of understanding. These two aspects are not only important from the perspective of developing computer applications for natural languages but also form the key components of language evolution and change.

Words take different senses in different contexts while appearing with other words. Context plays a vital role in disambiguation of word senses as well as in the interpretation of the actual meaning of words. For instance, the word "bank" has several distinct interpretations, including that of a "financial institution" and the "shore of a river." Automatic discovery and disambiguation of word senses from a given text is an important and challenging problem which has been extensively studied in the literature (Jones, 1986; Ide and Veronis, 1998; Schütze, 1998; Navigli, 2009). However, another equally important aspect that has not been so far well investigated corresponds to one or more changes that a word might undergo in its sense. This particular aspect is getting increasingly attainable as more and more time-varying text data become available in the form of millions of digitized books (Goldberg and Orwant, 2013) gathered over the last centuries. As a motivating example one could consider the word "sick" – while according to the standard English dictionaries the word is normally used to refer to some sort of illness, a new meaning of "sick" referring to something that is "crazy" or "cool" is currently getting popular in the English vernacular. This change is further interesting because while traditionally "sick" has been associated to something negative in general, the current meaning associates positivity with it. In fact, a rock band by the name of "Sick Puppies" has been founded which probably is inspired by the newer sense of the word sick. The title of this paper has been motivated by the above observation. Note that this phenomena of change in word senses has existed ever since the beginning of human communication (Bamman and Crane, 2011; Michel et

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1020–1029,
Baltimore, Maryland, USA, June 23-25 2014.

al., 2011; Wijaya and Yeniterzi, 2011; Mihalcea and Nastase, 2012); however, with the advent of modern technology and the availability of huge volumes of time-varying data it now has become possible to automatically track such changes and, thereby, help the lexicographers in word sense discovery, and design engineers in enhancing various NLP/IR applications (e.g., disambiguation, semantic search etc.) that are naturally sensitive to change in word senses.

The above motivation forms the basis of the central objective set in this paper, which is to devise a completely unsupervised approach to track noun sense changes in large texts available over multiple timescales. Toward this objective we make the following contributions: (a) devise a time-varying graph clustering based sense induction algorithm, (b) use the time-varying sense clusters to develop a split-join based approach for identifying new senses of a word, and (c) evaluate the performance of the algorithms on various datasets using different suitable approaches along with a detailed error analysis. Remarkably, comparison with the English WordNet indicates that in 44% cases, as identified by our algorithm, there has been a birth of a completely novel sense, in 46% cases a new sense has split off from an older sense and in 43% cases two or more older senses have merged in to form a new sense.

The remainder of the paper is organized as follows. In the next section we present a short review of the literature. In Section 3 we briefly describe the datasets and outline the process of co-occurrence graph construction. In Section 4 we present an approach based on graph clustering to identify the time-varying sense clusters and in Section 5 we present the split-merge based approach for tracking word sense changes. Evaluation methods are summarized in Section 6. Finally, conclusions and further research directions are outlined in Section 7.

2 Related work

Word sense disambiguation as well as word sense discovery have both remained key areas of research right from the very early initiatives in natural language processing research. Ide and Veronis (1998) present a very concise survey of the history of ideas used in word sense disambiguation; for a recent survey of the state-of-the-art one can refer to (Navigli, 2009). Some of the first attempts to automatic word sense discovery were made by Karen Spärck Jones (1986); later in lexicography, it has been extensively used as a pre-processing step for preparing mono- and multi-lingual dictionaries (Kilgarriff and Tugwell, 2001; Kilgarriff, 2004). However, as we have already pointed out that none of these works consider the temporal aspect of the problem.

In contrast, the current study, is inspired by works on language dynamics and opinion spreading (Mukherjee et al., 2011; Maity et al., 2012; Loreto et al., 2012) and automatic topic detection and tracking (Allan et al., 1998). However, our work differs significantly from those proposed in the above studies. Opinion formation deals with the self-organisation and emergence of shared vocabularies whereas our work focuses on how the different senses of these vocabulary words change over time and thus become "out-of-vocabulary". Topic detection involves detecting the occurrence of a new event such as a plane crash, a murder, a jury trial result, or a political scandal in a stream of news stories from multiple sources and tracking is the process of monitoring a stream of news stories to find those that track (or discuss) the same event. This is done on shorter timescales (hours, days), whereas our study focuses on larger timescales (decades, centuries) and we are interested in common nouns, verbs and adjectives as opposed to events that are characterized mostly by named entities. Other similar works on dynamic topic modelling can be found in (Blei and Lafferty, 2006; Wang and McCallum, 2006). Google books n-gram viewer[1] is a phrase-usage graphing tool which charts the yearly count of selected letter combinations, words, or phrases as found in over 5.2 million digitized books. It only reports frequency of word usage over the years, but does not give any correlation among them as e.g., in (Heyer et al., 2009), and does not analyze their senses.

A few approaches suggested by (Bond et al., 2009; Pääkkö and Lindén, 2012) attempt to augment WordNet synsets primarily using methods of annotation. Another recent work by Cook et al. (2013) attempts to induce word senses and then identify novel senses by comparing two different corpora: the "focus corpora" (i.e., a recent version of the corpora) and the "reference corpora" (older version of the corpora). However, this method is limited as it only considers two time points to

[1] https://books.google.com/ngrams

identify sense changes as opposed to our approach which is over a much larger timescale, thereby, effectively allowing us to track the points of change and the underlying causes. One of the closest work to what we present here has been put forward by (Tahmasebi et al., 2011), where the authors analyze a newspaper corpus containing articles between 1785 and 1985. The authors mainly report the frequency patterns of certain words that they found to be candidates for change; however a detailed cause analysis as to why and how a particular word underwent a sense change has not been demonstrated. Further, systematic evaluation of the results obtained by the authors has not been provided.

All the above points together motivated us to undertake the current work where we introduce, for the first time, a completely unsupervised and automatic method to identify the change of a word sense and the cause for the same. Further, we also present an extensive evaluation of the proposed algorithm in order to test its overall accuracy and performance.

3 Datasets and graph construction

In this section, we outline a brief description of the dataset used for our experiments and the graph construction procedure. The primary source of data have been the millions of digitized books made available through the Google Book project (Goldberg and Orwant, 2013). The Google Book syntactic n-grams dataset provides dependency fragment counts by the years. However, instead of using the plain syntactic n-grams, we use a far richer representation of the data in the form of a distributional thesaurus (Lin, 1997; Rychlý and Kilgarriff, 2007). In specific, we prepare a distributional thesaurus (DT) for each of the time periods separately and subsequently construct the required networks. We briefly outline the procedure of thesauri construction here referring the reader to (Riedl and Biemann, 2013) for further details. In this approach, we first extract each word and a set of its context features, which are formed by labeled and directed dependency parse edges as provided in the dataset. Following this, we compute the frequencies of the word, the context and the words along with their context. Next we calculate the lexicographer's mutual information LMI (Kilgarriff, 2004) between a word and its features and retain only the top 1000 ranked features for every word. Finally, we construct the DT network as follows: each word is a node in the network and the edge weight between two nodes is defined as the number of features that the two corresponding words share in common.

4 Tracking sense changes

The basic idea of our algorithm for tracking sense changes is as follows. If a word undergoes a sense change, this can be detected by comparing its senses obtained from two different time periods. Since we aim to detect this change automatically, we require distributional representations corresponding to word senses for different time periods. We, therefore, utilize the basic hypothesis of unsupervised sense induction to induce the sense clusters over various time periods and then compare these clusters to detect sense change. The basic premises of the 'unsupervised sense induction' are briefly described below.

4.1 Unsupervised sense induction

We use the co-occurrence based graph clustering framework introduced in (Biemann, 2006). The algorithm proceeds in three basic steps. Firstly, a co-occurrence graph is created for every target word found in DT. Next, the neighbourhood/ego graph is clustered using the Chinese Whispers (CW) algorithm (see (McAuley and Leskovec, 2012) for similar approaches). The algorithm, in particular, produces a set of clusters for each target word by decomposing its open neighborhood. We hypothesize that each different cluster corresponds to a particular sense of the target word. For a detailed description, the reader is referred to (Biemann, 2011).

If a word undergoes sense change, this can be detected by comparing the sense clusters obtained from two different time periods by the algorithm outlined above. For this purpose, we use statistics from the DT corresponding to two different time intervals, say tv_i and tv_j. We then run the sense induction algorithm over these two different datasets. Now, for a given word w that appears in both the datasets, we get two different set of clusters, say C_i and C_j. Without loss of generality, let us assume that our algorithm detects m sense clusters for the word w in tv_i and n sense clusters in tv_j. Let $C_i = \{s_{i1}, s_{i2}, \ldots, s_{im}\}$ and $C_j = \{s_{j1}, s_{j2}, \ldots, s_{jn}\}$, where s_{kz} denotes z^{th} sense cluster for word w during time interval tv_k.

We next describe our algorithm for detecting sense change from these sets of sense clusters.

4.2 Split, join, birth and death

We hypothesize that word w can undergo sense change from one time interval (tv_i) to another (tv_j) as per one of the following scenarios:

Split A sense cluster s_{iz} in tv_i *splits* into two (or more) sense clusters, s_{jp_1} and s_{jp_2} in tv_j

Join Two sense clusters s_{iz_1} and s_{iz_2} in tv_i *join* to make a single cluster s_{jp} in tv_j

Birth A new sense cluster s_{jp} appears in tv_j, which was absent in tv_i

Death A sense cluster s_{iz} in tv_i dies out and does not appear in tv_j

To detect split, join, birth or death, we build an $(m+1) \times (n+1)$ matrix I to capture the intersection between sense clusters of two different time periods. The first m rows and n columns correspond to the sense clusters in tv_i and tv_j espectively. We append an additional row and column to capture the fraction of words, which did not show up in any of the sense clusters in another time interval. So, an element I_{kl} of the matrix

- $1 \leq k \leq m, 1 \leq l \leq n$: denotes the fraction of words in a newer sense cluster s_{jl}, that were also present in an older sense cluster s_{ik}.
- $k = m+1, 1 \leq l \leq n$: denotes the fraction of words in the sense cluster s_{jl}, that were not present in any of the m clusters in tv_i.
- $1 \leq k \leq m, l = n+1$: denotes the fraction of words in the sense cluster s_{ik}, that did not show up in any of the n clusters in tv_j.

Thus, the matrix I captures all the four possible scenarios for sense change. Since we can not expect a perfect split, birth etc., we used certain threshold values to detect if a candidate word is undergoing sense change via one of these four cases. In Figure 1, as an example, we illustrate the birth of a new sense for the word 'compiler'.

4.3 Multi-stage filtering

To make sure that the candidate words obtained via our algorithm are meaningful, we applied multi-stage filtering to prune the candidate word list. The following criterion were used for the filtering:

Stage 1 We utilize the fact that the CW algorithm is non-deterministic in nature. We apply CW three times over the source and target time intervals. We obtain the candidate word lists using our algorithm for the three runs, then take the intersection to output those words, which came up in all the three runs.

Stage 2 From the above list, we retain only those candidate words, which have a part-of-speech tag 'NN' or 'NNS', as we focus on nouns for this work.

Stage 3 We sort the candidate list obtained in Stage 2 as per their occurrence in the first time period. Then, we remove the top 20% and the bottom 20% words from this list. Therefore, we consider the *torso* of the frequency distribution which is the most informative part for this type of an analysis.

5 Experimental framework

For our experiments, we utilized DTs created for 8 different time periods: 1520-1908, 1909-1953, 1954-1972, 1973-1986, 1987-1995, 1996-2001, 2002-2005 and 2006-2008 (Riedl et al., 2014). The time periods were set such that the amount of data in each time period is roughly the same. We will also use T_1 to T_8 to denote these time periods. The parameters for CW clustering were set as follows. The size of the neighbourhood (N) to be clustered was set to 200. The parameter n regulating the edge density in this neighbourhood was set to 200 as well. The parameter a was set to lin, which corresponds to favouring smaller clusters by hub downweighing[2]. The threshold values used to detect the sense changes were as follows. For birth, at least 80% words of the target cluster should be novel. For split, each split cluster should have at least 30% words of the source cluster and the total intersection of all the split clusters should be $> 80\%$. The same parameters were used for the join and death case with the interchange of source and target clusters.

5.1 Signals of sense change

Making comparisons between all the pairs of time periods gave us 28 candidate words lists. For

[2]data available at `http://sf.net/p/jobimtext/wiki/LREC2014_Google_DT/`

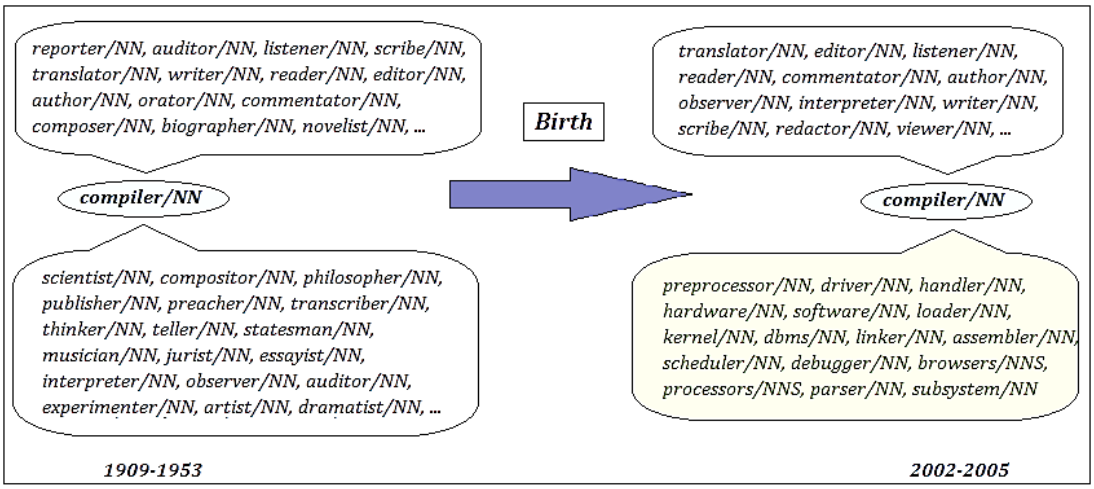


Figure 1: Example of the birth of a new sense for the word 'compiler'

each of these comparison, we applied the multi-stage filtering to obtain the pruned list of candidate words. Table 1 provides some statistics about the number of candidate words obtained corresponding to the birth case. The rows correspond to the source time-period and the columns correspond to the target time periods. An element of the table shows the number of candidate words obtained by comparing the corresponding source and target time periods.

Table 1: Number of candidate birth senses between all time periods

	T_2	T_3	T_4	T_5	T_6	T_7	T_8
T_1	2498	3319	3901	4220	4238	4092	3578
T_2		1451	2330	2789	2834	2789	2468
T_3			917	1460	1660	1827	1815
T_4				517	769	1099	1416
T_5					401	818	1243
T_6						682	1107
T_7							609

The table clearly shows a trend. For most of the cases, the number of candidate birth senses tends to increase as we go from left to right. Similarly, this number decreases as we go down in the table. This is quite intuitive since going from left to right corresponds to increasing the gap between two time periods while going down corresponds to decreasing this gap. As the gap increases (decreases), one would expect more (less) new senses coming in. Even while moving diagonally, the candidate words tend to decrease as we move downwards. This corresponds to the fact that the number of years in the time periods decreases as we move downwards, and therefore, the gap also decreases.

5.2 Stability analysis & sense change location

Formally, we consider a sense change from tv_i to tv_j **stable** if it was also detected while comparing tv_i with the following time periods tv_ks. This number of subsequent time periods, where the same sense change is detected, helps us to determine the **age** of a new sense. Similarly, for a candidate sense change from tv_i to tv_j, we say that the **location** of the sense change is tv_j if and only if that sense change does not get detected by comparing tv_i with any time interval tv_k, intermediate between tv_i and tv_j.

Table 1 gives a lot of candidate words for sense change. However, not all the candidate words were stable. Thus, it was important to prune these results using stability analysis. Also, it is to be noted that these results do not pin-point to the exact time-period, when the sense change might have taken place. For instance, among the 4238 candidate birth sense detected by comparing T_1 and T_6, many of these new senses might have come up in between T_2 to T_5 as well. We prune these lists further based on the stability of the sense, as well as to locate the approximate time interval, in which the sense change might have occurred.

Table 2 shows the number of stable (at least twice) senses as well as the number of stable sense changes located in that particular time period. While this decreases recall, we found this to be beneficial for the accuracy of the method.

Once we were able to locate the senses as well as to find the age of the senses, we attempted to

Table 2: Number of candidate birth senses obtained for different time periods

	T_2	T_3	T_4	T_5	T_6	T_7
T_1	2498	3319	3901	4220	4238	4092
stable	537	989	1368	1627	1540	1299
located	537	754	772	686	420	300
T_2		1451	2330	2789	2834	2789
stable		343	718	938	963	810
located		343	561	517	357	227

select some representative words and plotted them on a timeline as per the birth period and their age in Figure 2. The source time period here is 1909-1953.

6 Evaluation framework

During evaluation, we considered the clusters obtained using the 1909-1953 time-slice as our reference and attempted to track sense change by comparing these with the clusters obtained for 2002-2005. The sense change detected was categorized as to whether it was a new sense (birth), a single sense got split into two or more senses (split) or two or more senses got merged (join) or a particular sense died (death). We present a few instances of the resulting clusters in the paper and refer the reader to the supplementary material[3] for the rest of the results.

6.1 Manual evaluation

The algorithm detected a lot of candidate words for the cases of birth, split/join as well as death. Since it was difficult to go through all the candidate sense changes for all the comparisons manually, we decided to randomly select some candidate words, which were flagged by our algorithm as undergoing sense change, while comparing 1909-1953 and 2002-2005 DT. We selected 48 random samples of candidate words for birth cases and 21 random samples for split/join cases. One of the authors annotated each of the birth cases identifying whether or not the algorithm signalled a true sense change while another author did the same task for the split/join cases. The accuracy as per manual evaluation was found to be 60.4% for the birth cases and 57% for the split/join cases.

Table 3 shows the evaluation results for a few candidate words, flagged due to birth. Columns correspond to the candidate words, words obtained in the cluster of each candidate word (we will use the term 'birth cluster' for these words, henceforth), which indicated a new sense, the results of manual evaluation as well as the possible sense this birth cluster denotes.

Table 4 shows the corresponding evaluation results for a few candidate words, flagged due to split or join.

A further analysis of the words marked due to birth in the random samples indicates that there are 22 technology-related words, 2 slangs, 3 economics related words and 2 general words. For the split-join case we found that there are 3 technology-related words while the rest of the words are general. Therefore one of the key observations is that most of the technology related words (where the neighborhood is completely new) could be extracted from our birth results. In contrast, for the split-join instances most of the results are from the general category since the neighborhood did not change much here; it either got split or merged from what it was earlier.

6.2 Automated evaluation with WordNet

In addition to manual evaluation, we also performed automated evaluation for the candidate words. We chose WordNet for automated evaluation because not only does it have a wide coverage of word senses but also it is being maintained and updated regularly to incorporate new senses. We did this evaluation for the candidate birth, join and split sense clusters obtained by comparing 1909-1953 time period with respect to 2002-2005. For our evaluation, we developed an aligner to align the word clusters obtained with WordNet senses. The aligner constructs a WordNet dictionary for the purpose of synset alignment. The CW cluster is then aligned to WordNet synsets by comparing the clusters with WordNet graph and the synset with the maximum alignment score is returned as the output. In summary, the aligner tool takes as input the CW cluster and returns a WordNet synset id that corresponds to the cluster words. The evaluation settings were as follows:

Birth: For a candidate word flagged as birth, we first find out the set of all WordNet synset ids for its CW clusters in the source time period (1909-1953 in this case). Let S_{init} denote the union of these synset ids. We then find WordNet synset id for its birth-cluster, say s_{new}. Then, if $s_{new} \notin$

[3] http://cse.iitkgp.ac.in/resgrp/cnerg/acl2014_wordsense/

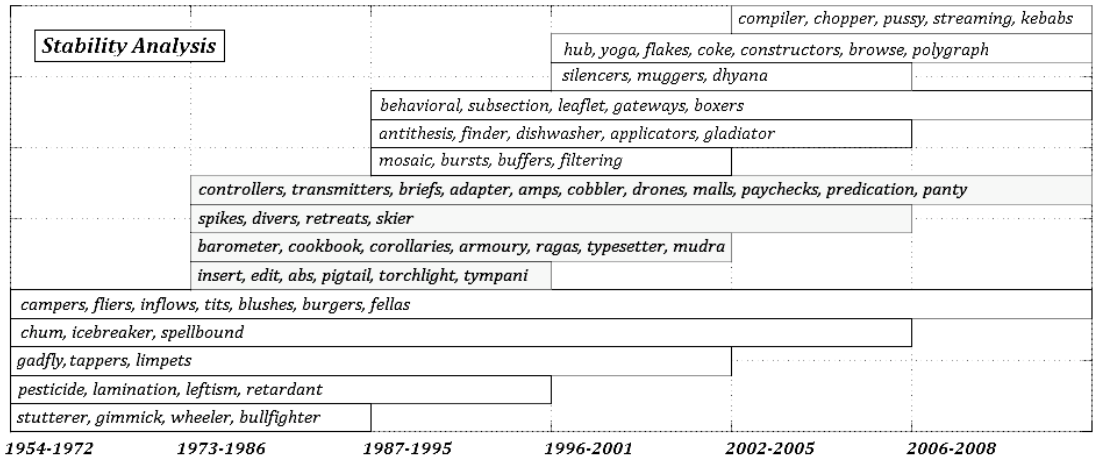


Figure 2: Examples of birth senses placed on a timeline as per their location as well as age

Table 3: Manual evaluation for seven randomly chosen candidate birth clusters between time periods 1909-1953 and 2002-2005

Sl No.	Candidate Word	birth cluster	Evaluation judgement, comments
1	implant	*gel, fibre, coatings, cement, materials, metal, filler silicone, composite, titanium, polymer, coating*	**No**, New set of words but similar sense already existed
2	passwords	*browsers, server, functionality, clients, workstation printers, software, protocols, hosts, settings, utilities*	**Yes**, New sense related to 'a computer sense'
3	giants	*multinationals, conglomerates, manufacturers corporations, competitors, enterprises, companies businesses, brands, firms*	**Yes**, New sense as 'an organization with very great size or force'
4	donation	*transplantation, donation, fertilization, transfusions transplant, transplants, insemination, donors, donor ...*	**Yes**, The new usage of donation associated with body organs etc.
5	novice	*negro, fellow, emigre, yankee, realist, quaker, teen male, zen, lady, admiring, celebrity, thai, millionaire ...*	**No**, this looks like a false positive
6	partitions	*server, printers, workstation, platforms, arrays modules, computers, workstations, kernel ...*	**Yes**, New usage related to the 'computing' domain
7	yankees	*athletics, cubs, tigers, sox, bears, braves, pirates cardinals, dodgers, yankees, giants, cardinals ...*	**Yes**, related to the 'New York Yankees' team

S_{init}, it implies that this is a new sense that was not present in the source clusters and we call it a 'success' as per WordNet.

Join: For the join case, we find WordNet synset ids s_1 and s_2 for the clusters obtained in the source time period and s_{new} for the join cluster in the target time period. If $s_1 \neq s_2$ and s_{new} is either s_1 or s_2, we call it a 'success'.

Split: For the split case, we find WordNet synset id s_{old} for the source cluster and synset ids s_1 and s_2 for the target split clusters. If $s_1 \neq s_2$ and either s_1, or s_2 retains the id s_{old}, we call it a 'success'.

Table 5 show the results of WordNet based evaluation. In case of birth we observe a success of 44% while for split and join we observe a success of 46% and 43% respectively. We then manually verified some of the words that were deemed as successes, as well as investigated WordNet sense they were mapped to. Table 6 shows some of the words for which the evaluation detected success along with WordNet senses. Clearly, the cluster words correspond to a newer sense for these words

Table 5: Results of the automatic evaluation using WordNet

Category	No. of Candidate Words	Success Cases
Birth	810	44%
Split	24	46%
Join	28	43%

Table 4: Manual evaluation for five randomly chosen candidate split/join clusters between time periods 1909-1953 and 2002-2005

Sl No.	Candidate Word	Source and target clusters
1	intonation (**split**)	S: *whisper, glance, idioms, gesture, chant, sob, inflection, diction, sneer, rhythm, accents ...* T_1: *nod, tone, grimace, finality, gestures, twang, shake, shrug, irony, scowl, twinkle ...* T_2: *accents, phrase, rhythm, style, phonology, diction, utterance, cadence, harmonies ...*
	Yes, T_1 corresponds to intonation in normal conversations while T_2 corresponds to the use of accents in formal and research literature	
2	diagonal (**split**)	S: *coast, edge, shoreline, coastline, border, surface, crease, edges, slope, sides, seaboard ...* T_1: *circumference, center, slant, vertex, grid, clavicle, margin, perimeter, row, boundary ..* T_2: *border, coast, seaboard, seashore, shoreline, waterfront, shore, shores, coastline, coasts*
	Yes, the split T_1 is based on mathematics where as T_2 is based on geography	
3	mantra (**join**)	S_1: *sutra, stanza, chanting, chants, commandments, monologue, litany, verse, verses ...* S_2: *praise, imprecation, benediction, praises, curse, salutation, benedictions, eulogy ...* T: *blessings, spell, curses, spells, rosary, prayers, blessing, prayer, benediction ...*
	Yes, the two seemingly distinct senses of mantra - a contextual usage for chanting and prayer (S_1) and another usage in its effect - salutations, benedictions (S_2) have now merged in T.	
4	continuum (**split**)	S: *circumference, ordinate, abscissa, coasts, axis, path, perimeter, arc, plane axis ...* T_1: *roadsides, corridors, frontier, trajectories, coast, shore, trail, escarpment, highways ...* T_2: *arc, ellipse, meridians, equator, axis, axis, plane, abscissa, ordinate, axis, meridian ....*
	Yes, the split S_1 denotes the usage of 'continuum' with physical objects while the the split S_2 corresponds to its usages in mathematics domain.	
5	headmaster (**join**)	S_1: *master, overseer, councillor, chancellor, tutors, captain, general, principal ...* S_2: *mentor, confessor, tutor, founder, rector, vicar, graduate, counselor, lawyer ...* T: *chaplain, commander, surveyor, coordinator, consultant, lecturer, inspector ...*
	No, it seems a false positive	

and the mapped WordNet synset matches the birth cluster to a very high degree.

6.3 Evaluation with a slang list

Slangs are words and phrases that are regarded as very informal, and are typically restricted to a particular context. New slang words come up every now and then, and this plays an integral part in the phenomena of sense change. We therefore decided to perform an evaluation as to how many slang words were being detected by our candidate birth clusters. We used a list of slangs available from the slangcity website[4]. We collected slangs for the years 2002-2005 and found the intersection with our candidate birth words. Note that the website had a large number of multi-word expressions that we did not consider in our study. Further, some of the words appeared as either erroneous or very transient (not existing more than a few months) entires, which had to be removed from the list. All these removal left us with a very little space for comparison; however, despite this we found 25 slangs from the website that were present in our birth results, e.g. 'bum', 'sissy', 'thug', 'dude' etc.

[4] http://slangcity.com/email_archive/index_2003.htm

6.4 Evaluation of candidate death clusters

Much of our evaluation was focussed on the birth sense clusters, mainly because these are more interesting from a lexicographic perspective. Additionally, the main theme of this work was to detect new senses for a given word. To detect a true death of a sense, persistence analysis was required, that is, to verify if the sense was persisting earlier and vanished after a certain time period. While such an analysis goes beyond the scope of this paper, we selected some interesting candidate "death" senses. Table 7 shows some of these interesting candidate words, their death cluster along with the possible vanished meaning, identified by the authors. While these words are still used in a related sense, the original meaning does not exist in the modern usage.

7 Conclusions

In this paper, we presented a completely unsupervised method to detect word sense changes by analyzing millions of digitized books archived spanning several centuries. In particular, we constructed DT networks over eight different time windows, clustered these networks and compared these clusters to identify the emergence of novel

Table 6: Example of randomly chosen candidate birth clusters mapped to WordNet

Sl No.	Candidate Word	birth cluster	Synset Id, WordNet sense
1	macro	*code, query, handler, program, procedure, subroutine module, script*	**6582403**, a set sequence of steps, part of larger computer program
2	caller	*browser, compiler, sender, routers, workstation, cpu host, modem, router, server*	**4175147**, a computer that provides client stations with access to files
3	searching	*coding, processing, learning, computing, scheduling planning, retrieval, routing, networking, navigation*	**1144355**, programming: setting an order and time for planned events
4	hooker	*bitch, whore, stripper, woman slut, prostitute girl, dancer ...*	**10485440**, a woman who engages in sexual intercourse for money
5	drones	*helicopters, fighters, rockets, flights, planes vehicles, bomber, missions, submarines ...*	**4264914**, a craft capable of traveling in outer space
6	amps	*inverters, capacitor, oscillators, switches, mixer transformer, windings, capacitors, circuits ...*	**2955247**, electrical device characterized by its capacity to store an electric charge
7	compilers	*interfaces, algorithms, programming, software modules, libraries, routines, tools, utilities ...*	**6566077**, written programs pertaining to the operation of a computer system

Table 7: Some representative examples for candidate death sense clusters

Sl No.	Candidate Word	death cluster	Vanished meaning
1	slop	*jeans, velveteen, tweed, woollen, rubber, sealskin, wear oilskin, sheepskin, velvet, calico, deerskin, goatskin, cloth ...*	clothes and bedding supplied to sailors by the navy
2	blackmail	*subsidy, rent, presents, tributes, money, fine, bribes dues, tolls, contributions, contribution, customs, duties ...*	Origin: denoting protection money levied by Scottish chiefs
3	repertory	*dictionary, study, compendium, bibliography, lore, directory catalogues, science, catalog, annals, digest, literature ...*	Origin: denoting an index or catalog: from late Latin repertorium
4	phrasing	*contour, outline, construction, handling, grouping, arrangement structure, modelling, selection, form ...*	in the sense 'style or manner of expression': via late Latin Greek phrasis

senses. The performance of our method has been evaluated manually as well as by comparison with WordNet and a list of slang words. Through manual evaluation we found that the algorithm could correctly identify 60.4% birth cases from a set of 48 random samples and 57% split/join cases from a set of 21 randomly picked samples. Quite strikingly, we observe that (i) in 44% cases the birth of a novel sense is attested by WordNet, (ii) in 46% cases the split of an older sense is signalled on comparison with WordNet and (iii) in 43% cases the join of two senses is attested by WordNet. These results might have strong lexicographic implications – even if one goes by very moderate estimates almost half of the words would be candidate entries in WordNet if they were not already part of it. This method can be extremely useful in the construction of lexico-semantic networks for low-resource languages, as well as for keeping lexico-semantic resources up to date in general.

Future research directions based on this work are manifold. On one hand, our method can be used by lexicographers in designing new dictionaries where candidate new senses can be semi-automatically detected and included, thus greatly reducing the otherwise required manual effort. On the other hand, this method can be directly used for various NLP/IR applications like semantic search, automatic word sense discovery as well as disambiguation. For semantic search, taking into account the newer senses of the word can increase the relevance of the query result. Similarly, a disambiguation engine informed with the newer senses of a word can increase the efficiency of disambiguation, and recognize senses uncovered by the inventory that would otherwise have to be wrongly assigned to covered senses. In addition, this method can be also extended to the 'NNP' part-of-speech (i.e., named entities) to identify changes in role of a person/place. Furthermore, it would be interesting to apply this method to languages other than English and to try to align new senses of cognates across languages.

Acknowledgements

AM would like to thank DAAD for supporting the faculty exchange programme to TU Darmstadt. PG would like to thank Google India Private Ltd. for extending travel support to attend the conference. MR and CB have been supported by an IBM SUR award and by LOEWE as part of the research center *Digital Humanities*.

References

J. Allan, R. Papka and V. Lavrenko. 1998. On-line new event detection and tracking. In proceedings of *SIGIR*, 37–45, Melbourne, Australia.

D. Bamman and G. Crane. 2011. Measuring Historical Word Sense Variation. In proceedings of *JCDL*, 1–10, New York, NY, USA.

C. Biemann. 2006. Chinese whispers - an efficient graph clustering algorithm and its application to natural language processing problems. In proceedings of *TextGraphs*, 73–80, New York, USA.

C. Biemann. 2011. *Structure Discovery in Natural Language*. Springer Heidelberg Dordrecht London New York. ISBN 978-3-642-25922-7.

D. Blei and J. Lafferty. 2006. Dynamic topic models. In proceedings of *ICML*, 113–120, Pittsburgh, Pennsylvania.

F. Bond, H. Isahara, S. Fujita, K. Uchimoto, T. Kuribayash and K. Kanzaki. 2009. Enhancing the Japanese WordNet. In proceedings of *workshop on Asian Language Resources*, 1–8, Suntec, Singapore.

P. Cook, J. H. Lau, M. Rundell, D. McCarthy, T. Baldwin. 2013. A lexicographic appraisal of an automatic approach for detecting new word senses. In proceedings of *eLex*, 49-65, Tallinn, Estonia.

Y. Goldberg and J. Orwant. 2013. A dataset of syntactic-ngrams over time from a very large corpus of English books. In proceedings of the *Joint Conference on Lexical and Computational Semantics (*SEM)*, 241–247, Atlanta, GA, USA.

G. Heyer, F. Holz and S. Teresniak. 2009. Change of topics over time – tracking topics by their change of meaning. In proceedings of *KDIR*, Madeira, Portugal.

N. Ide and J. Veronis. 1998. Introduction to the special issue on word sense disambiguation: The state of the art. *Computational Linguistics*, 24(1):1–40.

A. Kilgarriff, P. Rychly, P. Smrz, and D. Tugwell. 2004. The sketch engine. In Proceedings of *EURALEX*, 105–116, Lorient, France.

A. Kilgarriff and D. Tugwell. 2001. Word sketch: Extraction and display of significant collocations for lexicography. In proceedings of *COLLOCATION: Computational Extraction, Analysis and Exploitation*, 32–38, Toulouse, France.

D. Lin. 1997. Using syntactic dependency as local context to resolve word sense ambiguity. In proceedings of *ACL/EACL*, 64–71, Madrid, Spain.

V. Loreto, A. Mukherjee and F. Tria. 2012. On the origin of the hierarchy of color names. *PNAS*, 109(18), 6819–6824.

S. K. Maity, T. M. Venkat and A. Mukherjee. 2012. Opinion formation in time-varying social networks: The case of the naming game. *Phys. Rev. E*, 86, 036110.

J. McAuley and J. Leskovec. 2012. Learning to discover social circles in ego networks. In proceedings of *NIPS*, 548–556, Nevada, USA.

J.-B. Michel, Y. K. Shen, A. P. Aiden, A. Veres, M. K. Gray, J. P. Pickett, D. Hoiberg, D. Clancy, P. Norvig, J. Orwant, S. Pinker, M. A. Nowak and E. L. Aiden. 2011. Quantitative analysis of culture using millions of digitized books. *Science*, 331(6014):176–182.

R. Mihalcea and V. Nastase. 2012. Word epoch disambiguation: finding how words change over time. In proceedings of *ACL*, 259–263, Jeju Island, Korea.

A. Mukherjee, F. Tria, A. Baronchelli, A. Puglisi and V. Loreto. 2011. Aging in language dynamics. *PLoS ONE*, 6(2): e16677.

R. Navigli. 2009. Word sense disambiguation: a survey. *ACM Computing Surveys*, 41(2):1–69.

P. Pääkkö and K. Lindén. 2012. Finding a location for a new word in WordNet. In proceedings of the *Global WordNet Conference*, Matsue, Japan.

M. Riedl and C. Biemann. 2013. Scaling to large^3 data: An efficient and effective method to compute distributional thesauri. In proceedings of *EMNLP*, 884–890, Seattle, Washington, USA.

M. Riedl, R. Steuer and C. Biemann. 2014. Distributed distributional similarities of Google books over the centuries. In proceedings of *LREC*, Reykjavik, Iceland.

P. Rychlý and A. Kilgarriff. 2007. An efficient algorithm for building a distributional thesaurus (and other sketch engine developments). In proceedings of *ACL, poster and demo sessions*, 41–44, Prague, Czech Republic.

H. Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123.

K. Spärk-Jones. 1986. *Synonymy and Semantic Classification*. Edinburgh University Press. ISBN 0-85224-517-3.

N. Tahmasebi, T. Risse and S. Dietze. 2011. Towards automatic language evolution tracking: a study on word sense tracking. In proceedings of *EvoDyn*, vol. 784, Bonn, Germany.

X. Wang and A. McCallum. 2006. Topics over time: a non-Markov continuous-time model of topical trends. In proceedings of *KDD*, 424–433, Philadelphia, PA, USA.

D. Wijaya and R. Yeniterzi. 2011. Understanding semantic change of words over centuries. In proceedings of the workshop on *Detecting and Exploiting Cultural Diversity on the Social Web*, 35–40, Glasgow, Scotland, UK.

A Step-wise Usage-based Method for Inducing Polysemy-aware Verb Classes

Daisuke Kawahara[†] **Daniel W. Peterson**[‡] **Martha Palmer**[‡]
[†]Kyoto University, Kyoto, Japan
[‡]University of Colorado at Boulder, Boulder, CO, USA
dk@i.kyoto-u.ac.jp, {Daniel.W.Peterson, Martha.Palmer}@colorado.edu

Abstract

We present an unsupervised method for inducing verb classes from verb uses in giga-word corpora. Our method consists of two clustering steps: verb-specific semantic frames are first induced by clustering verb uses in a corpus and then verb classes are induced by clustering these frames. By taking this step-wise approach, we can not only generate verb classes based on a massive amount of verb uses in a scalable manner, but also deal with verb polysemy, which is bypassed by most of the previous studies on verb clustering. In our experiments, we acquire semantic frames and verb classes from two giga-word corpora, the larger comprising 20 billion words. The effectiveness of our approach is verified through quantitative evaluations based on polysemy-aware gold-standard data.

1 Introduction

A verb plays a primary role in conveying the meaning of a sentence. Capturing the sense of a verb is essential for natural language processing (NLP), and thus lexical resources for verbs play an important role in NLP.

Verb classes are one such lexical resource. Manually-crafted verb classes have been developed, such as Levin's classes (Levin, 1993) and their extension, VerbNet (Kipper-Schuler, 2005), in which verbs are organized into classes on the basis of their syntactic and semantic behavior. Such verb classes have been used in many NLP applications that need to consider semantics in particular, such as word sense disambiguation (Dang, 2004), semantic parsing (Swier and Stevenson, 2005; Shi and Mihalcea, 2005) and discourse parsing (Subba and Di Eugenio, 2009).

There have also been many attempts to automatically acquire verb classes with the goal of either adding frequency information to an existing resource or of inducing similar verb classes for other languages. Most of these approaches assume that all target verbs are monosemous (Stevenson and Joanis, 2003; Schulte im Walde, 2006; Joanis et al., 2008; Li and Brew, 2008; Sun et al., 2008; Sun and Korhonen, 2009; Vlachos et al., 2009; Parisien and Stevenson, 2010; Parisien and Stevenson, 2011; Falk et al., 2012; Lippincott et al., 2012; Reichart and Korhonen, 2013; Sun et al., 2013). This monosemous assumption, however, is not realistic because many frequent verbs actually have multiple senses. Moreover, to the best of our knowledge, none of the following approaches attempt to quantitatively evaluate soft clusterings of verb classes induced by polysemy-aware unsupervised approaches (Korhonen et al., 2003; Lapata and Brew, 2004; Li and Brew, 2007; Schulte im Walde et al., 2008).

In this paper, we propose an unsupervised method for inducing verb classes that is aware of verb polysemy. Our method consists of two clustering steps: verb-specific semantic frames are first induced by clustering verb uses in a corpus and then verb classes are induced by clustering these frames. By taking this step-wise approach, we can not only induce verb classes with frequency information from a massive amount of verb uses in a scalable manner, but also deal with verb polysemy.

Our novel contributions are summarized as follows:

- induce both semantic frames and verb classes from a massive amount of verb uses by a scalable method,
- explicitly deal with verb polysemy,
- discover effective features for each of the clustering steps, and
- quantitatively evaluate a soft clustering of verbs.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1030–1040,
Baltimore, Maryland, USA, June 23-25 2014.

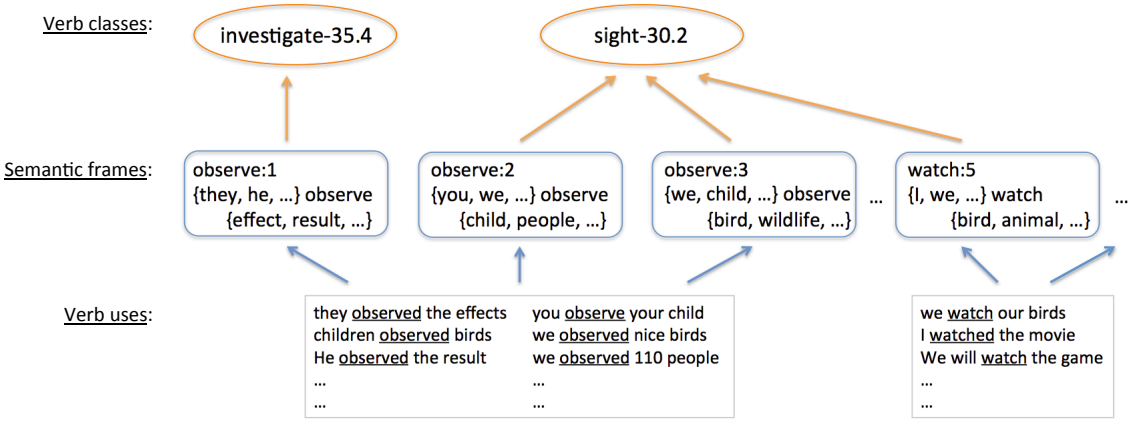


Figure 1: Overview of our two-step approach. Verb-specific semantic frames are first induced from verb uses (lower part) and then verb classes are induced from the semantic frames (upper part). The labels of verb classes are manually assigned here for better understanding.

2 Related Work

As stated in Section 1, most of the previous studies on verb clustering assume that verbs are monosemous. A typical method in these studies is to represent each verb as a single data point and apply classification (e.g., Joanis et al. (2008)) or clustering (e.g., Sun and Korhonen (2009)) to these data points. As a representation for a data point, distributions of subcategorization frames are often used, and other semantic features (e.g., selectional preferences) are sometimes added to improve the performance.

Among these studies on monosemous verb clustering (i.e., predominant class induction), there have been several Bayesian methods. Vlachos et al. (2009) proposed a Dirichlet process mixture model (DPMM; Neal (2000)) to cluster verbs based on subcategorization frame distributions. They evaluated their result with a gold-standard test set, where a single class is assigned to a verb. Parisien and Stevenson (2010) proposed a hierarchical Dirichlet process (HDP; Teh et al. (2006)) model to jointly learn argument structures (subcategorization frames) and verb classes by using syntactic features. Parisien and Stevenson (2011) extended their model by adding semantic features. They tried to account for verb learning by children and did not evaluate the resultant verb classes. Modi et al. (2012) extended the model of Titov and Klementiev (2012), which is an unsupervised model for inducing semantic roles, to jointly induce semantic roles and frames across verbs using the Chinese Restaurant Process (Aldous, 1985). All of the above methods considered verbs to be monosemous and did not deal with verb polysemy. Our approach also uses Bayesian methods, but is designed to capture verb polysemy.

We summarize a few studies that consider polysemy of verbs in the rest of this section.

Miyao and Tsujii (2009) proposed a supervised method that can handle verb polysemy. Their method represents a verb's syntactic and semantic features, and learns a log-linear model from the SemLink corpus (Loper et al., 2007). Boleda et al. (2007) also proposed a supervised method for Catalan adjectives considering the polysemy of adjectives.

The most closely related work to our polysemy-aware task of unsupervised verb class induction is the work of Korhonen et al. (2003), who used distributions of subcategorization frames to cluster verbs. They adopted the Nearest Neighbor (NN) and Information Bottleneck (IB) methods for clustering. In particular, they tried to consider verb polysemy by using the IB method, which is a soft clustering method (Tishby et al., 1999). However, the verb itself is still represented as a single data point. After performing soft clustering, they noted that most verbs fell into a single class, and they decided to assign a single class to each verb by hardening the clustering. They considered multiple classes only in the gold-standard data used for their evaluations. We also evaluate our induced verb classes on this gold-standard data, which was created on the basis of Levin's classes (Levin, 1993).

Lapata and Brew (2004) and Li and Brew (2007) proposed probabilistic models for calculating prior probabilities of verb classes for a verb. These models are approximated to condition not

on verbs but on subcategorization frames. As mentioned in Li and Brew (2007), it is desirable to extend the model to depend on verbs to further improve accuracy. They conducted several evaluations including predominant class induction and token-level verb sense disambiguation, but did not evaluate multiple classes output by their models. Schulte im Walde et al. (2008) also applied probabilistic soft clustering to verbs by incorporating subcategorization frames and selectional preferences based on WordNet. This model is based on the Expectation-Maximization algorithm and the Minimum Description Length principle. Since they focused on the incorporation of selectional preferences, they did not evaluate verb classes but evaluated only selectional preferences using a language model-based measure.

Materna proposed LDA-frames, which are defined across verbs and can be considered to be a kind of verb class (Materna, 2012; Materna, 2013). LDA-frames are probabilistic semantic frames automatically induced from a raw corpus. He used a model based on latent Dirichlet allocation (LDA; Blei et al. (2003)) and the Dirichlet process to cluster verb instances of a triple (subject, verb, object) to produce semantic frames and roles. Both of these are represented as a probabilistic distribution of words across verbs. He applied this method to the BNC and acquired 1,200 frames and 400 roles (Materna, 2012). He did not evaluate the resulting frames as verb classes.

In sum, there have been no studies that quantitatively evaluate polysemous verb classes automatically induced by unsupervised methods.

3 Our Approach

3.1 Overview

Our objective is to automatically learn semantic frames and verb classes from a massive amount of verb uses following usage-based approaches. Although Bayesian approaches are a possible solution to simultaneously induce frames and verb classes from a corpus as used in previous studies, it has prohibitive computational cost. For instance, Parisien and Stevenson applied HDP only to a small-scale child speech corpus that contains 170K verb uses to jointly induce subcategorization frames and verb classes (Parisien and Stevenson, 2010; Parisien and Stevenson, 2011). Materna applied an LDA-based method to the BNC, which contains 1.4M verb uses, to induce semantic frames across verbs that can be considered to be verb classes (Materna, 2012; Materna, 2013). However, it would take three months for this experiment using this 100 million word corpus.[1] Although it is best to use the largest possible corpus for this kind of knowledge acquisition tasks (Sasano et al., 2009), it is infeasible to scale to giga-word corpora using such joint models.

In this paper, we propose a two-step approach for inducing semantic frames and verb classes. First, we make multiple data points for each verb to deal with verb polysemy (cf. polysemy-aware previous studies still represented a verb as one data point (Korhonen et al., 2003; Miyao and Tsujii, 2009)). To do that, we induce verb-specific semantic frames by clustering verb uses. Then, we induce verb classes by clustering these verb-specific semantic frames across verbs. An interesting point here is that we can use exactly the same method for these two clustering steps.

Our procedure to automatically induce verb classes from verb uses is summarized as follows:

1. induce verb-specific semantic frames by clustering predicate-argument structures for each verb extracted from automatic parses as shown in the lower part of Figure 1, and
2. induce verb classes by clustering the induced semantic frames across verbs as shown in the upper part of Figure 1.

Each of these two steps is described in the following sections in detail.

3.2 Inducing Verb-specific Semantic Frames

We induce verb-specific semantic frames from verb uses based on the method of Kawahara et al. (2014). Our semantic frames consist of case slots, each of which consists of word instances that can be filled. The procedure for inducing these semantic frames is as follows:

1. apply dependency parsing to a raw corpus and extract predicate-argument structures for each verb from the automatic parses,
2. merge the predicate-argument structures that have presumably the same meaning based on the assumption of one sense per collocation (Yarowsky, 1993) to get a set of initial frames, and

[1] In our replication experiment, it took a week to perform 70 iterations using Materna's code and an Intel Xeon E5-2680 (2.7GHz) CPU. To reach 1,000 iterations, which are reported to be optimum, it would take three months.

3. apply clustering to the initial frames based on the Chinese Restaurant Process (Aldous, 1985) to produce verb-specific semantic frames.

These three steps are briefly described below.

3.2.1 Extracting Predicate-argument Structures from a Raw Corpus

We apply dependency parsing to a large raw corpus. We use the Stanford parser with Stanford dependencies (de Marneffe et al., 2006).[2] Collapsed dependencies are adopted to directly extract prepositional phrases.

Then, we extract predicate-argument structures from the dependency parses. Dependents that have the following dependency relations to a verb are extracted as arguments:

> nsubj, xsubj, dobj, iobj, ccomp, xcomp, prep_$*$

In this process, the verb and arguments are lemmatized, and only the head of an argument is preserved for compound nouns.

Predicate-argument structures are collected for each verb and the subsequent processes are applied to the predicate-argument structures of each verb.

3.2.2 Constructing Initial Frames from Predicate-argument Structures

To make the computation feasible, we merge the predicate-argument structures that have the same or similar meaning to get initial frames. These initial frames are the input of the subsequent clustering process. For this merge, we assume one sense per collocation (Yarowsky, 1993) for predicate-argument structures.

For each predicate-argument structure of a verb, we couple the verb and an argument to make a unit for sense disambiguation. We select an argument in the following order by considering the degree of effect on the verb sense:[3]

> dobj, ccomp, nsubj, prep_$*$, iobj.

Then, the predicate-argument structures that have the same verb and argument pair (slot and word, e.g., "dobj:effect") are merged into an initial frame. After this process, we discard minor initial frames that occur fewer than 10 times.

[2] http://nlp.stanford.edu/software/lex-parser.shtml

[3] If a predicate-argument structure has multiple prepositional phrases, one of them is randomly selected.

3.2.3 Clustering Method

We cluster initial frames for each verb to produce semantic frames using the Chinese Restaurant Process (Aldous, 1985), regarding each initial frame as an instance.

We calculate the posterior probability of a cluster c_j given an initial frame f_i as follows:

$$P(c_j|f_i) \propto \begin{cases} \frac{n(c_j)}{N+\alpha} \cdot P(f_i|c_j) & c_j \neq new \\ \frac{\alpha}{N+\alpha} \cdot P(f_i|c_j) & c_j = new, \end{cases} \quad (1)$$

where N is the number of initial frames for the target verb and $n(c_j)$ is the current number of initial frames assigned to the cluster c_j. α is a hyper-parameter that determines how likely it is for a new cluster to be created. In this equation, the first term is the Dirichlet process prior and the second term is the likelihood of f_i.

$P(f_i|c_j)$ is defined based on the Dirichlet-Multinomial distribution as follows:

$$P(f_i|c_j) = \prod_{w \in V} P(w|c_j)^{count(f_i,w)}, \quad (2)$$

where V is the vocabulary in all case slots cooccurring with the verb and $count(f_i, w)$ is the number of w in the initial frame f_i. The original method in Kawahara et al. (2014) defined w as pairs of slots and words, e.g., "nsubj:child" and "dobj:bird," but does not consider slot-only features, e.g., "nsubj" and "dobj," which ignore lexical information. Here we experiment with both representations and compare the results.

$P(w|c_j)$ is defined as follows:

$$P(w|c_j) = \frac{count(c_j, w) + \beta}{\sum_{t \in V} count(c_j, t) + |V| \cdot \beta}, \quad (3)$$

where $count(c_j, w)$ is the current number of w in the cluster c_j, and β is a hyper-parameter of Dirichlet distribution. For a new cluster, this probability is uniform ($1/|V|$).

We regard each output cluster as a semantic frame, by merging the initial frames in a cluster into a semantic frame. In this way, semantic frames for each verb are acquired.

We use Gibbs sampling to realize this clustering.

3.3 Inducing Verb Classes from Semantic Frames

To induce verb classes across verbs, we apply clustering to the induced verb-specific semantic

frames. We can use exactly the same clustering method as described in Section 3.2.3 by using semantic frames for multiple verbs as an input instead of initial frames for a single verb. This is because an initial frame has the same structure as a semantic frame, which is produced by merging initial frames. We regard each output cluster as a verb class this time.

For the features, w, in equation (2), we try the two representations again: slot-only features and slot-word pair features. The representation using only slots corresponds to the consideration of only syntactic argument patterns. The other representation using the slot-word pairs means that semantic similarity based on word overlap is naturally considered by looking at lexical information. We will compare in our experiments four possible combinations: two feature representations for each of the two clustering steps.

4 Experiments and Evaluations

We first describe our experimental settings and define evaluation metrics to evaluate induced soft clusterings of verb classes. Then, we conduct type-level multi-class evaluations, type-level single-class evaluations and token-level multi-class evaluations. These two levels of evaluations are performed by considering the work of Reichart et al. (2010) on clustering evaluation. Finally, we discuss the results of our full experiments.

4.1 Experimental Settings

We use two kinds of large-scale corpora: a web corpus and the English Gigaword corpus.

To prepare a web corpus, we extracted sentences from crawled web pages that are judged to be written in English based on the encoding information. Then, we selected sentences that consist of at most 40 words, and removed duplicated sentences. From this process, we obtained a corpus of one billion sentences, totaling approximately 20 billion words. We focused on verbs whose frequency in the web corpus was more than 1,000. There were 19,649 verbs, including phrasal verbs, and separating passive and active constructions. We extracted 2,032,774,982 predicate-argument structures.

We also used the English Gigaword corpus (LDC2011T07; English Gigaword Fifth Edition). This corpus consists of approximately 180 million sentences, which totaling four billion words. There were 7,356 verbs after applying the same frequency threshold as the web corpus. We extracted 423,778,278 predicate-argument structures from this corpus.

We set the hyper-parameters α in (1) and β in (3) to 1.0. The cluster assignments for all the components were initialized randomly. We took 100 samples for each input frame and selected the cluster assignment that has the highest probability.

4.2 Evaluation Metrics

To measure the precision and recall of a clustering, modified purity and inverse purity (also called collocation or weighted class accuracy) are commonly used in previous studies on verb clustering (e.g., Sun and Korhonen (2009)). However, since these measures are only applicable to a hard clustering, it is necessary to extend them to be applicable to a soft clustering, because in our task a verb can belong to multiple clusters or classes.[4] We propose a normalized version of modified purity and inverse purity. This kind of normalization for soft clusterings was performed for other evaluation metrics as in Springorum et al. (2013).

To measure the precision of a clustering, a normalized version of modified purity is defined as follows. Suppose K is the set of automatically induced clusters and G is the set of gold classes. Let K_i be the verb vector of the i-th cluster and G_j be the verb vector of the j-th gold class. Each component of these vectors is a normalized frequency, which equals a cluster/class attribute probability given a verb. Where there is no frequency information available for class distribution, such as the gold-standard data described in Section 4.3, we use a uniform distribution across the verb's classes. The core idea of purity is that each cluster K_i is associated with its most prevalent gold class. In addition, to penalize clusters that consist of only one verb, such singleton clusters in K are considered as errors, as is usual with modified purity. The normalized modified purity (nmPU) can then be written as follows:

$$\text{nmPU} = \frac{1}{N} \sum_{i \text{ s.t. } |K_i|>1} \max_j \delta_{K_i}(K_i \cap G_j), \quad (4)$$

$$\delta_{K_i}(K_i \cap G_j) = \sum_{v \in K_i \cap G_j} c_{iv}, \quad (5)$$

[4] Korhonen et al. (2003) evaluated hard clusterings based on a gold standard with multiple classes per verb. They reported only precision measures including modified purity, and avoided extending the evaluation metrics for soft clusterings.

verb	classes	verb	classes
place	**9**	drop	**9**, 45, 004, 47,
dye	**24**, 21, 41		51, A54, A30
focus	**31**, 45	bake	**26**, 45
stare	**30**	persuade	**002**
lay	**9**	sparkle	**43**
build	**26**, 45	pour	**9**, 43, 26, 57,
force	**002**, 11		13, 31
glow	**43**	invent	**26**, 27

Table 1: An excerpt of the gold-standard verb classes for several verbs from Korhonen et al. (2003). The classes starting with '0' were derived from the LCS database, those starting with 'A' were defined by Korhonen et al., and the other classes were from Levin's classes. A bolded class is the predominant class for each verb.

where N denotes the total number of verbs, $|K_i|$ denotes the number of positive components in K_i, and c_{iv} denotes the v-th component of K_i. $\delta_{K_i}(K_i \cap G_j)$ means the total mass of the set of verbs in $K_i \cap G_j$, given by summing up the values in K_i. In case of evaluating a hard clustering, this is equal to $|K_i \cap G_j|$ because all the values of c_{iv} are equal to 1.

As usual, the following normalized inverse purity (niPU) is used to measure the recall of a clustering:

$$\text{niPU} = \frac{1}{N} \sum_j \max_i \delta_{G_j}(K_i \cap G_j). \quad (6)$$

Finally, we use the harmonic mean (F_1) of nmPU and niPU as a single measure of clustering quality.

4.3 Type-level Multi-class Evaluations

We first evaluate our induced verb classes on the test set created by Korhonen et al. (2003) (Table 1 of their paper) which was created by considering verb polysemy on the basis of Levin's classes and the LCS database (Dorr, 1997). It consists of 62 classes and 110 verbs, out of which 35 verbs are monosemous and 75 verbs are polysemous. The average number of verb classes per verb is 2.24. An excerpt from this data is shown in Table 1.

As our baselines, we adopt two previously proposed methods. We first implemented a soft clustering method for verb class induction proposed by Korhonen et al. (2003). They used the information bottleneck (IB) method for assigning probabilities of classes to each verb. Note that Korhonen et al. (2003) actually hardened the clusterings and left the evaluations of soft clusterings for their future work. For input data, we employ VALEX (Korhonen et al., 2006), which is a publicly-available large-scale subcategorization lexicon.[5] By following the method of Korhonen et al. (2003), prepositional phrases (pp) are parameterized for two frequent subcategorization frames (NP and NP_PP), and the unfiltered raw frequencies of subcategorization frames are used as features to represent a verb. It is necessary to specify the number of clusters, k, for the IB method beforehand, and we adopt 35 and 42 clusters according to their reported high accuracies. To output multiple classes for each verb, we set a threshold, t, for class attribute probabilities. That is, classes that have a higher class attribute probability than the threshold are output for each verb. We report the results of the following threshold values: 0.01, 0.02, 0.05 and 0.10.

The other baseline is LDA-frames (Materna, 2012). We use the induced LDA-frames that are

method	K	nmPU	niPU	F_1
IB (k=35, t=0.10)	35.0	53.59	51.44	52.44
IB (k=35, t=0.05)	35.0	53.67	52.62	53.10
IB (k=35, t=0.02)	35.0	54.42	54.43	54.40
IB (k=35, t=0.01)	35.0	54.60	55.54	55.04
IB (k=42, t=0.10)	41.6	55.42	49.46	52.24
IB (k=42, t=0.05)	41.8	55.55	49.97	52.59
IB (k=42, t=0.02)	42.0	56.19	51.24	53.58
IB (k=42, t=0.01)	42.0	56.80	51.92	54.24
LDA-frames (t=0.10)	100	47.52	56.83	51.76
LDA-frames (t=0.05)	165	50.46	67.94	57.91
LDA-frames (t=0.02)	306	49.98	75.50	60.14
LDA-frames (t=0.01)	458	49.55	82.71	61.97
Gigaword/S-S	272.8	63.46	67.66	65.49
Gigaword/S-SW	36.4	31.49	95.70	47.38
Gigaword/SW-S	186.2	63.52	64.18	63.84
Gigaword/SW-SW	30.0	36.27	94.66	52.40
web/S-S	363.6	61.32	78.64	68.90
web/S-SW	52.2	35.80	**99.30**	52.62
web/SW-S	212.2	**66.26**	77.38	**71.39**
web/SW-SW	55.0	36.70	96.25	53.13

Table 2: Type-level multi-class evaluations. K represents the (average) number of induced classes. "S" denotes the use of slot-only features and "SW" denotes the use of slot-word pair features. For example, "SW-S" means that slot-word pair features are used for semantic frame induction and slot-only features are used for verb class induction.

[5] http://ilexir.co.uk/applications/valex/

method	K	predominant class eval mPU	iPU	F_1	multiple class eval mPU	niPU	F_1
NN	24	46.36	52.73	49.34	52.73	46.85	49.62
IB (k=35)	34.8	42.73	51.82	46.82	51.64	46.83	49.09
IB (k=42)	41.0	**47.45**	50.91	49.11	**55.27**	45.45	49.87
LDA-frames	53	30.00	47.27	36.71	41.82	44.28	43.01
Gigaword/S	9.6	25.64	71.27	37.70	32.91	64.71	43.62
Gigaword/SW	10.6	30.36	71.09	42.25	39.82	66.92	49.70
web/S	20.4	42.73	61.46	**50.31**	54.91	57.12	55.86
web/SW	11.8	34.36	**71.82**	46.40	49.09	**67.01**	**56.50**

Table 3: Type-level single-class evaluations against predominant/multiple classes. K represents the (average) number of induced classes.

available on the web site.[6] This frame data was induced from the BNC and consists of 1,200 frames and 400 semantic roles. Again, we set a threshold for frame attribute probabilities.

We report results using our methods with four feature combinations (slot-only (S) and slot-word pair (SW) features each used for both the frame-generation and verb-class clustering steps) for both the Gigaword and web corpora. Table 2 lists evaluation results for the baseline methods and our methods.[7] The results of the IB baseline and our methods are obtained by averaging five runs.

We can see that "web/SW-S" achieved the best performance and obtained a higher F_1 than the baselines by more than nine points. "Web/SW-S" uses the combination of slot-word pair features for clustering verb-specific frames and slot-only features for clustering across verbs. Interestingly, this result indicates that slot distributions are more effective than lexical information in slot-word pairs for inducing verb classes similar to the gold standard. This result is consistent with expectations, given a gold standard based on Levin's verb classes, which are organized according to the syntactic behavior of verbs. The use of slot-word pairs for verb class induction generally merged too many frames into each class, apparently due to accidental word overlaps across verbs.

The verb classes induced from the web corpus achieved a higher F_1 than those from the Gigaword corpus. This can be attributed to the larger size of the web corpus. The employment of this kind of huge corpus is enabled by our scalable method.

[6] http://nlp.fi.muni.cz/projekty/lda-frames/

[7] Although we do not think that the classes with very small attribute probabilities are meaningful, the F_1 scores for lower thresholds than 0.01 converged to about 66 in the case of LDA-frames.

4.4 Type-level Single-class Evaluations against Predominant/Multiple Classes

Since we focus on the handling of verb polysemy, predominant class induction for each verb is not our main objective. However, we wish to compare our method with previous work on the induction of a predominant (monosemous) class for each verb.

To output a single class for each verb by using our proposed method, we skip the induction of verb-specific semantic frames and instead create a single frame for each verb by merging all predicate-argument structures of the verb. Then, we apply clustering to these frames across verbs. For clustering features, we again compare two representations: slot-only features (S) and slot-word pair features (SW).

We evaluate the single-class output for each verb based on the predominant gold-standard classes, which are defined for each verb in the test set of Korhonen et al. (2003). This data contains 110 verbs and 33 classes. We evaluate these single-class outputs in the same manner as Korhonen et al. (2003), using the gold standard with multiple classes, which we also use for our multi-class evaluations.

As we did with the multi-class evaluations, we adopt modified purity (mPU), inverse purity (iPU) and their harmonic mean (F_1) as the metrics for the evaluation with predominant classes. It is not necessary to normalize these metrics when we treat verbs as monosemous, and evaluate against the predominant sense. When we evaluate against the multiple classes in the gold standard, we do normalize the inverse purity.

For baselines, we once more adopt the Nearest Neighbor (NN) and Information Bottleneck (IB) methods proposed by Korhonen et al. (2003), and LDA-frames proposed by Materna (2012). The

clusterings with the NN and IB methods are obtained by using the VALEX subcategorization lexicon. To harden the clusterings of the IB method and the LDA-frames, the class with the highest probability is selected for each verb. This hardening process is exactly the same as Korhonen et al. (2003). Note that our results of the NN and IB methods are different from those reported in their paper since the data source is different.[8]

Table 3 lists accuracies of baseline methods and our methods. Our proposed method using the web corpus achieved comparable performance with the baseline methods on the predominant class evaluation and outperformed them on the multiple class evaluation. More sophisticated methods for predominant class induction, such as the method of Sun and Korhonen (2009) using selectional preferences, could produce better single-class outputs, but have difficulty in producing polysemy-aware verb classes.

From the result, we can see that the induced verb classes based on slot-only features did not achieve a higher F_1 than those based on slot-word pair features in many cases. This result is different from that of multi-class evaluations in Section 4.3. We speculate that slot distributions are not so different among verbs when all uses of a verb are merged into one frame, and thus their discrimination power is lower than that in the intermediate construction of semantic frames.

4.5 Token-level Multi-class Evaluations

We conduct token-level multi-class evaluations using 119 verbs, which appear 100 or more times in sections 02-21 of the SemLink WSJ corpus. These 119 verbs cover 102 VerbNet classes, and 48 of them are polysemous in the sense of being in more than one VerbNet class. Each instance of these 119 verbs in this corpus belongs to one of 102 VerbNet classes. We first add these instances to the instances from a raw corpus and apply the two-step clustering to these merged instances. Then, we compare the induced verb classes of the SemLink instances with their gold-standard VerbNet classes. We report the values of modified purity (mPU), inverse purity (iPU) and their harmonic mean (F_1). It is not necessary to normalize these metrics because the clustering of these instances is hard.

[8] Korhonen et al. (2003) reported that the highest modified purity was 49% against predominant classes and 60% against multiple classes.

method	K	mPU	iPU	F_1
Gigaword/S-NIL	–	93.43	20.06	33.03
Gigaword/SW-NIL	–	94.45	41.07	57.25
Gigaword/S-S	512.2	75.06	45.26	56.47
Gigaword/SW-S	260.6	73.98	56.45	64.04
web/S-NIL	–	93.70	32.96	48.76
web/SW-NIL	–	**94.51**	44.95	60.92
web/S-S	500.0	72.25	52.48	60.79
web/SW-S	255.2	72.65	**61.00**	**66.31**

Table 4: Token-level evaluations against VerbNet classes. K represents the average number of induced classes.

For clustering features, we compare two feature combinations: "S-S" and "SW-S," which achieved high performance in the type-level multi-class evaluations (Section 4.3). The results of these methods are obtained by averaging five runs. For a baseline, we use verb-specific semantic frames without clustering across verbs ("S-NIL" and "SW-NIL"), where these frames are considered to be verb classes but not shared across verbs. Table 4 lists accuracies of these methods for the two corpora. We can see that "SW-S" achieved a higher F_1 than "S-S" and the baselines without verb class induction ("S-NIL" and "SW-NIL").

Modi et al. (2012) induced semantic frames across verbs using the monosemous assumption and reported an F_1 of 44.7% (77.9% PU and 31.4% iPU) for the assignment of FrameNet frames to the FrameNet corpus. We also conducted the above evaluation against FrameNet frames for 75 verbs.[9] We achieved an F_1 of 62.79% (66.97% mPU and 59.09% iPU) for "web/SW-S," and an F_1 of 60.06% (65.58% mPU and 55.39% iPU) for "Gigaword/SW-S." It is difficult to directly compare these results with Modi et al. (2012), but our induced verb classes seem to have higher F_1 accuracy.

4.6 Full Experiments and Discussions

We finally induce verb classes from the semantic frames of 1,667 verbs, which appear at least once in sections 02-21 of the WSJ corpus. Based on the best results in the above evaluations, we induced semantic frames using slot-word pair features, and then induced verb classes using slot-only features. We ended with 38,481 semantic frames and 699 verb classes from the Gigaword

[9] Since FrameNet frames are not assigned to all verbs of SemLink, the number of verbs is different from the evaluations against VerbNet classes.

class	semantic frames
Class 1	rave:1, talk:1
Class 2	need:2, say:2
Class 3	smell:1, sound:1
Class 4	concentrate:1, focus:1
Class 5	express:2, inquire:62, voice:1
Class 6	revolve:1, snake:2, wrap:2
Class 7	hand:1, hand:3, hand:4
Class 8	depend:1, rely:1, rely:3
Class 9	collaborate:1, compete:2, work:1
Class 10	coach:3, teach:3, teach:4
Class 11	dance:1, react:1, stick:1
Class 12	advise:8, express:4, quiz:10, voice:2
Class 13	give:18, grant:6, offer:11, offer:12
Class 14	keep:14, keep:18, stay:4, stay:488
Class 15	cuff:5, fasten:2, tie:1, tie:4
Class 16	arrange:3, book:4, make:27, reserve:5
Class 17	deport:6, differ:1, fluctuate:1, vary:1
Class 18	peek:1, peek:3, peer:1, peer:7, ...
Class 19	groan:1, growl:1, hiss:1, moan:1, purr:1
Class 20	inform:1, notify:2, remind:1, beware:1, ...

Table 5: Examples of induced verb classes. Underlined semantic frames are shown in Table 6.

	slot	instance words
need:2	nsubj	you:2150273, i:7678, we:4599, ...
	ccomp	⟨s⟩:2193321
say:2	nsubj	she:1705781, he:20693, i:9422, ...
	ccomp	⟨s⟩:1829616
inform:1	nsubj	i:11100, he:10323, we:6373, ...
	dobj	me:30646, you:27678, us:21642, ...
	prep_of	decision:846, this:759, situation:688, ...
	⋮	
notify:2	nsubj	we:7505, you:3439, i:1035, ...
	dobj	you:18604, us:7281, them:3649, ...
	prep_of	change:1540, problem:496, status:386, ...
	⋮	

Table 6: Examples of induced semantic frames. The number following an instance word denotes its frequency and ⟨s⟩ denotes a sentential complement.

corpus, and 61,903 semantic frames and 840 verb classes from the web corpus. It took two days to induce verb classes from the Gigaword corpus and three days from the web corpus.

Examples of verb classes and semantic frames induced from the web corpus are shown in Table 5 and Table 6. While there are many classes with consistent meanings, such as "Class 4" and "Class 16," some classes have mixed meanings. For instance, "Class 2" consists of the semantic frames "need:2" and "say:2." These frames were merged due to the high syntactic similarity of constituting slot distributions, which are comprised of a subject and a sentential complement. To improve the quality of verb classes, it is necessary to develop a clustering model that can consider syntactic and lexical similarity in a balanced way.

5 Conclusion

We presented a step-wise unsupervised method for inducing verb classes from instances in gigaword corpora. This method first clusters predicate-argument structures to induce verb-specific semantic frames and then clusters these semantic frames across verbs to induce verb classes. Both clustering steps are performed with exactly the same method, which is based on the Chinese Restaurant Process. The resulting semantic frames and verb classes are open to the public and also can be searched via our web interface.[10]

From the results, we can see that the combination of the slot-word pair features for clustering verb-specific frames and the slot-only features for clustering across verbs is the most effective and outperforms the baselines by approximately 10 points. This indicates that slot distributions are more effective than lexical information in slot-word pairs for the induction of verb classes, when Levin-style classes are used for evaluation. This is consistent with Levin's principle of organizing verb classes according to the syntactic behavior of verbs.

As applications of the resulting semantic frames and verb classes, we plan to integrate them into syntactic parsing, semantic role labeling and verb sense disambiguation. For instance, Kawahara and Kurohashi (2006) improved accuracy of dependency parsing based on Japanese semantic frames automatically induced from a raw corpus. It is also valuable and promising to apply the induced verb classes to NLP applications as used in metaphor identification (Shutova et al., 2010) and argumentative zoning (Guo et al., 2011).

Acknowledgments

This work was supported by Kyoto University John Mung Program and JST CREST. We also gratefully acknowledge the support of the National Science Foundation Grant NSF-IIS-1116782, A Bayesian Approach to Dynamic Lexical Resources for Flexible Language Processing. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

[10] http://nlp.ist.i.kyoto-u.ac.jp/member/kawahara/cf/crp.en/

References

David Aldous. 1985. Exchangeability and related topics. *École d'Été de Probabilités de Saint-Flour XIII —1983*, pages 1–198.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *the Journal of Machine Learning Research*, 3:993–1022.

Gemma Boleda, Sabine Schulte im Walde, and Toni Badia. 2007. Modelling polysemy in adjective classes by multi-label classification. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 171–180.

Hoa Trang Dang. 2004. *Investigations into the role of lexical semantics in word sense disambiguation*. Ph.D. thesis, University of Pennsylvania.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, pages 449–454.

Bonnie J. Dorr. 1997. Large-scale dictionary construction for foreign language tutoring and interlingual machine translation. *Machine Translation*, 12(4):271–322.

Ingrid Falk, Claire Gardent, and Jean-Charles Lamirel. 2012. Classifying French verbs using French and English lexical resources. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 854–863.

Yufan Guo, Anna Korhonen, and Thierry Poibeau. 2011. A weakly-supervised approach to argumentative zoning of scientific documents. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 273–283.

Eric Joanis, Suzanne Stevenson, and David James. 2008. A general feature space for automatic verb classification. *Natural Language Engineering*, 14(3):337–367.

Daisuke Kawahara and Sadao Kurohashi. 2006. A fully-lexicalized probabilistic model for Japanese syntactic and case structure analysis. In *Proceedings of the Human Language Technology Conference of the NAACL*, pages 176–183.

Daisuke Kawahara, Daniel W. Peterson, Octavian Popescu, and Martha Palmer. 2014. Inducing example-based semantic frames from a massive amount of verb uses. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*.

Karin Kipper-Schuler. 2005. *VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon*. Ph.D. thesis, University of Pennsylvania.

Anna Korhonen, Yuval Krymolowski, and Zvika Marx. 2003. Clustering polysemic subcategorization frame distributions semantically. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 64–71.

Anna Korhonen, Yuval Krymolowski, and Ted Briscoe. 2006. A large subcategorization lexicon for natural language processing applications. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, pages 345–352.

Mirella Lapata and Chris Brew. 2004. Verb class disambiguation using informative priors. *Computational Linguistics*, 30(1):45–73.

Beth Levin. 1993. *English verb classes and alternations: A preliminary investigation*. The University of Chicago Press.

Jianguo Li and Chris Brew. 2007. Disambiguating Levin verbs using untagged data. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*.

Jianguo Li and Chris Brew. 2008. Which are the best features for automatic verb classification. In *Proceedings of ACL-08: HLT*, pages 434–442.

Thomas Lippincott, Anna Korhonen, and Diarmuid Ó Séaghdha. 2012. Learning syntactic verb frames using graphical models. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 420–429.

Edward Loper, Szu-Ting Yi, and Martha Palmer. 2007. Combining lexical resources: mapping between PropBank and VerbNet. In *Proceedings of the 7th International Workshop on Computational Linguistics*.

Jiří Materna. 2012. LDA-frames: An unsupervised approach to generating semantic frames. In *Proceedings of the 13th International Conference CICLing 2012, Part I*, pages 376–387.

Jiří Materna. 2013. Parameter estimation for LDA-frames. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 482–486.

Yusuke Miyao and Jun'ichi Tsujii. 2009. Supervised learning of a probabilistic lexicon of verb semantic classes. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1328–1337.

Ashutosh Modi, Ivan Titov, and Alexandre Klementiev. 2012. Unsupervised induction of frame-semantic representations. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*, pages 1–7.

Radford M. Neal. 2000. Markov chain sampling methods for Dirichlet process mixture models. *Journal of computational and graphical statistics*, 9(2):249–265.

Christopher Parisien and Suzanne Stevenson. 2010. Learning verb alternations in a usage-based Bayesian model. In *Proceedings of the 32nd Annual Meeting of the Cognitive Science Society*.

Christopher Parisien and Suzanne Stevenson. 2011. Generalizing between form and meaning using learned verb classes. In *Proceedings of the 33rd Annual Meeting of the Cognitive Science Society*.

Roi Reichart and Anna Korhonen. 2013. Improved lexical acquisition through DPP-based verb clustering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 862–872.

Roi Reichart, Omri Abend, and Ari Rappoport. 2010. Type level clustering evaluation: New measures and a POS induction case study. In *Proceedings of the 14th Conference on Computational Natural Language Learning*, pages 77–87.

Ryohei Sasano, Daisuke Kawahara, and Sadao Kurohashi. 2009. The effect of corpus size on case frame acquisition for discourse analysis. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 521–529.

Sabine Schulte im Walde, Christian Hying, Christian Scheible, and Helmut Schmid. 2008. Combining EM training and the MDL principle for an automatic verb classification incorporating selectional preferences. In *Proceedings of ACL-08: HLT*, pages 496–504.

Sabine Schulte im Walde. 2006. Experiments on the automatic induction of German semantic verb classes. *Computational Linguistics*, 32(2):159–194.

Lei Shi and Rada Mihalcea. 2005. Putting pieces together: Combining FrameNet, VerbNet and WordNet for robust semantic parsing. In *Computational Linguistics and Intelligent Text Processing*, pages 100–111. Springer.

Ekaterina Shutova, Lin Sun, and Anna Korhonen. 2010. Metaphor identification using verb and noun clustering. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1002–1010.

Sylvia Springorum, Sabine Schulte im Walde, and Jason Utt. 2013. Detecting polysemy in hard and soft cluster analyses of German preposition vector spaces. In *Proceedings of the 6th International Joint Conference on Natural Language Processing*, pages 632–640.

Suzanne Stevenson and Eric Joanis. 2003. Semi-supervised verb class discovery using noisy features. In *Proceedings of the 7th Conference on Natural Language Learning*, pages 71–78.

Rajen Subba and Barbara Di Eugenio. 2009. An effective discourse parser that uses rich linguistic information. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 566–574.

Lin Sun and Anna Korhonen. 2009. Improving verb clustering with automatically acquired selectional preferences. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 638–647.

Lin Sun, Anna Korhonen, and Yuval Krymolowski. 2008. Automatic classification of English verbs using rich syntactic features. In *Proceedings of the 3rd International Joint Conference on Natural Language Processing*, pages 769–774.

Lin Sun, Diana McCarthy, and Anna Korhonen. 2013. Diathesis alternation approximation for verb clustering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, Short Papers*, pages 736–741.

Robert Swier and Suzanne Stevenson. 2005. Exploiting a verb lexicon in automatic semantic role labelling. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 883–890.

Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. 2006. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476).

Naftali Tishby, Fernando C. Pereira, and William Bialek. 1999. The information bottleneck method. In *Proceedings of the 37th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377.

Ivan Titov and Alexandre Klementiev. 2012. A Bayesian approach to unsupervised semantic role induction. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 12–22.

Andreas Vlachos, Anna Korhonen, and Zoubin Ghahramani. 2009. Unsupervised and constrained Dirichlet process mixture models for verb clustering. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, pages 74–82.

David Yarowsky. 1993. One sense per collocation. In *Proceedings of the Workshop on Human Language Technology*, pages 266–271.

Structured Learning for Taxonomy Induction with Belief Propagation

Mohit Bansal
TTI Chicago
mbansal@ttic.edu

David Burkett
Twitter Inc.
dburkett@twitter.com

Gerard de Melo
Tsinghua University
gdm@demelo.org

Dan Klein
UC Berkeley
klein@cs.berkeley.edu

Abstract

We present a structured learning approach to inducing hypernym taxonomies using a probabilistic graphical model formulation. Our model incorporates heterogeneous relational evidence about both hypernymy and siblinghood, captured by semantic features based on patterns and statistics from Web n-grams and Wikipedia abstracts. For efficient inference over taxonomy structures, we use loopy belief propagation along with a directed spanning tree algorithm for the core hypernymy factor. To train the system, we extract sub-structures of WordNet and discriminatively learn to reproduce them, using adaptive subgradient stochastic optimization. On the task of reproducing sub-hierarchies of WordNet, our approach achieves a 51% error reduction over a chance baseline, including a 15% error reduction due to the non-hypernym-factored sibling features. On a comparison setup, we find up to 29% relative error reduction over previous work on ancestor F1.

1 Introduction

Many tasks in natural language understanding, such as question answering, information extraction, and textual entailment, benefit from lexical semantic information in the form of types and hypernyms. A recent example is IBM's Jeopardy! system Watson (Ferrucci et al., 2010), which used type information to restrict the set of answer candidates. Information of this sort is present in term taxonomies (e.g., Figure 1), ontologies, and thesauri. However, currently available taxonomies such as WordNet are incomplete in coverage (Pennacchiotti and Pantel, 2006; Hovy et al., 2009), unavailable in many domains and languages, and time-intensive to create or extend manually. There has thus been considerable interest in building lexical taxonomies automatically.

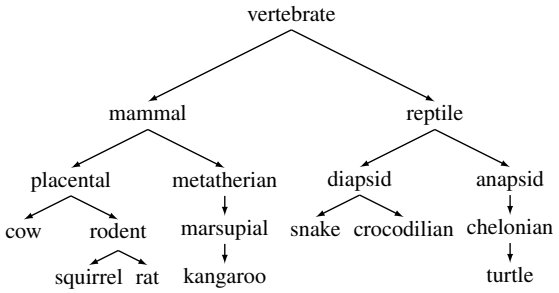


Figure 1: An excerpt of WordNet's vertebrates taxonomy.

In this work, we focus on the task of taking collections of terms as input and predicting a complete taxonomy structure over them as output. Our model takes a loglinear form and is represented using a factor graph that includes both 1st-order scoring factors on directed hypernymy edges (a parent and child in the taxonomy) and 2nd-order scoring factors on sibling edge pairs (pairs of hypernym edges with a shared parent), as well as incorporating a global (directed spanning tree) structural constraint. Inference for both learning and decoding uses structured loopy belief propagation (BP), incorporating standard spanning tree algorithms (Chu and Liu, 1965; Edmonds, 1967; Tutte, 1984). The belief propagation approach allows us to efficiently and effectively incorporate heterogeneous relational evidence via hypernymy and siblinghood (e.g., coordination) cues, which we capture by semantic features based on simple surface patterns and statistics from Web n-grams and Wikipedia abstracts. We train our model to maximize the likelihood of existing example ontologies using stochastic optimization, automatically learning the most useful relational patterns for full taxonomy induction.

As an example of the relational patterns that our

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1041–1051,
Baltimore, Maryland, USA, June 23-25 2014.

system learns, suppose we are interested in building a taxonomy for types of mammals (see Figure 1). Frequent attestation of hypernymy patterns like *rat is a rodent* in large corpora is a strong signal of the link *rodent* → *rat*. Moreover, sibling or coordination cues like *either rats or squirrels* suggest that *rat* is a sibling of *squirrel* and adds evidence for the links *rodent* → *rat* and *rodent* → *squirrel*. Our supervised model captures exactly these types of intuitions by automatically discovering such heterogeneous relational patterns as features (and learning their weights) on edges and on sibling edge pairs, respectively.

There have been several previous studies on taxonomy induction. e.g., the incremental taxonomy induction system of Snow et al. (2006), the longest path approach of Kozareva and Hovy (2010), and the maximum spanning tree (MST) approach of Navigli et al. (2011) (see Section 4 for a more detailed overview). The main contribution of this work is that we present the first discriminatively trained, structured probabilistic model over the full space of taxonomy trees, using a structured inference procedure through both the learning and decoding phases. Our model is also the first to directly learn relational patterns as part of the process of training an end-to-end taxonomic induction system, rather than using patterns that were hand-selected or learned via pairwise classifiers on manually annotated co-occurrence patterns. Finally, it is the first end-to-end (i.e., non-incremental) system to include sibling (e.g., coordination) patterns at all.

We test our approach in two ways. First, on the task of recreating fragments of WordNet, we achieve a 51% error reduction on ancestor-based F1 over a chance baseline, including a 15% error reduction due to the non-hypernym-factored sibling features. Second, we also compare to the results of Kozareva and Hovy (2010) by predicting the large *animal* subtree of WordNet. Here, we get up to 29% relative error reduction on ancestor-based F1. We note that our approach falls at a different point in the space of performance tradeoffs from past work – by producing complete, highly articulated trees, we naturally see a more even balance between precision and recall, while past work generally focused on precision.[1] To avoid presumption of a single optimal tradeoff, we also present results for precision-based decoding, where we trade off recall for precision.

2 Structured Taxonomy Induction

Given an input term set $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$, we wish to compute the conditional distribution over taxonomy trees $\boldsymbol{y}$. This distribution $P(\boldsymbol{y}|\boldsymbol{x})$ is represented using the graphical model formulation shown in Figure 2. A taxonomy tree $\boldsymbol{y}$ is composed of a set of indicator random variables y_{ij} (circles in Figure 2), where $y_{ij} = \text{ON}$ means that x_i is the parent of x_j in the taxonomy tree (i.e. there exists a directed edge from x_i to x_j). One such variable exists for each pair (i, j) with $0 \leq i \leq n$, $1 \leq j \leq n$, and $i \neq j$.[2]

In a factor graph formulation, a set of factors (squares and rectangles in Figure 2) determines the probability of each possible variable assignment. Each factor F has an associated scoring function ϕ_F, with the probability of a total assignment determined by the product of all these scores:

$$P(\boldsymbol{y}|\boldsymbol{x}) \propto \prod_F \phi_F(\boldsymbol{y}) \quad (1)$$

2.1 Factor Types

In the models we present here, there are three types of factors: EDGE factors that score individual edges in the taxonomy tree, SIBLING factors that score pairs of edges with a shared parent, and a global TREE factor that imposes the structural constraint that $\boldsymbol{y}$ form a legal taxonomy tree.

EDGE Factors. For each edge variable y_{ij} in the model, there is a corresponding factor E_{ij} (small blue squares in Figure 2) that depends only on y_{ij}. We score each edge by extracting a set of features $\mathbf{f}(x_i, x_j)$ and weighting them by the (learned) weight vector $\mathbf{w}$. So, the factor scoring function is:

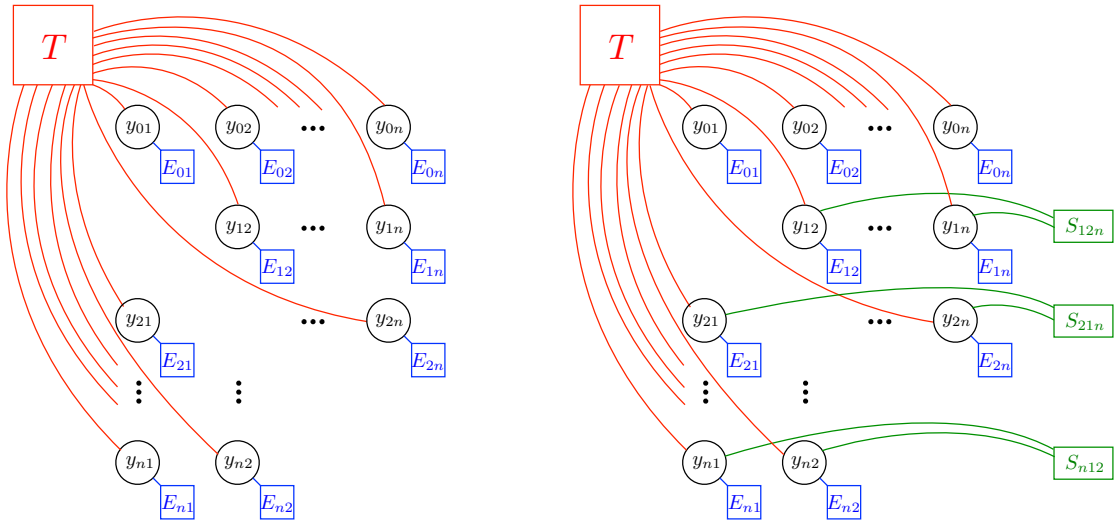
$$\phi_{E_{ij}}(y_{ij}) = \begin{cases} \exp(\mathbf{w} \cdot \mathbf{f}(x_i, x_j)) & y_{ij} = \text{ON} \\ \exp(0) = 1 & y_{ij} = \text{OFF} \end{cases}$$

SIBLING Factors. Our second model also includes factors that permit 2nd-order features looking at terms that are siblings in the taxonomy tree. For each triple (i, j, k) with $i \neq j$, $i \neq k$, and $j < k$,[3] we have a factor S_{ijk} (green rectangles in

[1] While different applications will value precision and recall differently, and past work was often intentionally precision-focused, it is certainly the case that an ideal solution would maximize both.

[2] We assume a special dummy root symbol x_0.

[3] The ordering of the siblings x_j and x_k doesn't matter here, so having separate factors for (i, j, k) and (i, k, j) would be redundant.



(a) Edge Features Only (b) Full Model

Figure 2: Factor graph representation of our model, both without (a) and with (b) SIBLING factors.

Figure 2b) that depends on y_{ij} and y_{ik}, and thus can be used to encode features that should be active whenever x_j and x_k share the same parent, x_i. The scoring function is similar to the one above:

$$\phi_{S_{ijk}}(y_{ij}, y_{ik}) = \begin{cases} \exp(\mathbf{w} \cdot \mathbf{f}(x_i, x_j, x_k)) & y_{ij} = y_{ik} = \text{ON} \\ 1 & \text{otherwise} \end{cases}$$

TREE Factor. Of course, not all variable assignments $\boldsymbol{y}$ form legal taxonomy trees (i.e., directed spanning trees). For example, the assignment $\forall i, j,\ y_{ij} = \text{ON}$ might get a high score, but would not be a valid output of the model. Thus, we need to impose a structural constraint to ensure that such illegal variable assignments are assigned 0 probability by the model. We encode this in our factor graph setting using a single global factor T (shown as a large red square in Figure 2) with the following scoring function:

$$\phi_T(\boldsymbol{y}) = \begin{cases} 1 & \boldsymbol{y} \text{ forms a legal taxonomy tree} \\ 0 & \text{otherwise} \end{cases}$$

Model. For a given global assignment $\boldsymbol{y}$, let

$$\mathbf{f}(\boldsymbol{y}) = \sum_{\substack{i,j \\ y_{ij}=\text{ON}}} \mathbf{f}(x_i, x_j) + \sum_{\substack{i,j,k \\ y_{ij}=y_{ik}=\text{ON}}} \mathbf{f}(x_i, x_j, x_k)$$

Note that by substituting our model's factor scoring functions into Equation 1, we get:

$$P(\boldsymbol{y}|\boldsymbol{x}) \propto \begin{cases} \exp(\mathbf{w} \cdot \mathbf{f}(\boldsymbol{y})) & \boldsymbol{y} \text{ is a tree} \\ 0 & \text{otherwise} \end{cases}$$

Thus, our model has the form of a standard loglinear model with feature function $\mathbf{f}$.

2.2 Inference via Belief Propagation

With the model defined, there are two main inference tasks we wish to accomplish: computing expected feature counts and selecting a particular taxonomy tree for a given set of input terms (decoding). As an initial step to each of these procedures, we wish to compute the marginal probabilities of particular edges (and pairs of edges) being on. In a factor graph, the natural inference procedure for computing marginals is belief propagation. Note that finding taxonomy trees is a structurally identical problem to directed spanning trees (and thereby non-projective dependency parsing), for which belief propagation has previously been worked out in depth (Smith and Eisner, 2008). Therefore, we will only briefly sketch the procedure here.

Belief propagation is a general-purpose inference method that computes marginals via directed messages passed from variables to adjacent factors (and vice versa) in the factor graph. These messages take the form of (possibly unnormalized) distributions over values of the variable. The two types of messages (variable to factor or factor to variable) have mutually recursive definitions. The message from a factor F to an adjacent variable V involves a sum over all possible values of every other variable that F touches. While the EDGE and SIBLING factors are simple enough to compute this sum by brute force, performing the sum naïvely for computing messages from the TREE factor would take exponential time. How-

ever, due to the structure of that particular factor, all of its outgoing messages can be computed simultaneously in $O(n^3)$ time via an efficient adaptation of Kirchhoff's Matrix Tree Theorem (MTT) (Tutte, 1984) which computes partition functions and marginals for directed spanning trees.

Once message passing is completed, marginal beliefs are computed by merely multiplying together all the messages received by a particular variable or factor.

2.2.1 Loopy Belief Propagation

Looking closely at Figure 2a, one can observe that the factor graph for the first version of our model, containing only EDGE and TREE factors, is acyclic. In this special case, belief propagation is exact: after one round of message passing, the beliefs computed (as discussed in Section 2.2) will be the true marginal probabilities under the current model. However, in the full model, shown in Figure 2b, the SIBLING factors introduce cycles into the factor graph, and now the messages being passed around often depend on each other and so they will change as they are recomputed. The process of iteratively recomputing messages based on earlier messages is known as *loopy* belief propagation. This procedure only finds *approximate* marginal beliefs, and is not actually guaranteed to converge, but in practice can be quite effective for finding workable marginals in models for which exact inference is intractable, as is the case here. All else equal, the more rounds of message passing that are performed, the closer the computed marginal beliefs will be to the true marginals, though in practice, there are usually diminishing returns after the first few iterations. In our experiments, we used a fairly conservative upper bound of 20 iterations, but in most cases, the messages converged much earlier than that.

2.3 Training

We used gradient-based maximum likelihood training to learn the model parameters $\mathbf{w}$. Since our model has a loglinear form, the derivative of $\mathbf{w}$ with respect to the likelihood objective is computed by just taking the gold feature vector and subtracting the vector of expected feature counts. For computing expected counts, we run belief propagation until completion and then, for each factor in the model, we simply read off the marginal probability of that factor being active (as computed in Section 2.2), and accumulate a partial count for each feature that is fired by that factor. This method of computing the gradient can be incorporated into any gradient-based optimizer in order to learn the weights $\mathbf{w}$. In our experiments we used AdaGrad (Duchi et al., 2011), an adaptive subgradient variant of standard stochastic gradient ascent for online learning.

2.4 Decoding

Finally, once the model parameters have been learned, we want to use the model to find taxonomy trees for particular sets of input terms. Note that if we limit our scores to be edge-factored, then finding the highest scoring taxonomy tree becomes an instance of the MST problem (also known as the maximum arborescence problem for the directed case), which can be solved efficiently in $O(n^2)$ quadratic time (Tarjan, 1977) using the greedy, recursive Chu-Liu-Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967).[4]

Since the MST problem can be solved efficiently, the main challenge becomes finding a way to ensure that our scores are edge-factored. In the first version of our model, we could simply set the score of each edge to be $\mathbf{w} \cdot \mathbf{f}(x_i, x_j)$, and the MST recovered in this way would indeed be the highest scoring tree: $\arg\max_{\boldsymbol{y}} P(\boldsymbol{y}|\boldsymbol{x})$. However, this straightforward approach doesn't apply to the full model which also uses sibling features. Hence, at decoding time, we instead start out by once more using belief propagation to find marginal beliefs, and then set the score of each edge to be its belief odds ratio: $\frac{b_{Y_{ij}}(\text{ON})}{b_{Y_{ij}}(\text{OFF})}$.[5]

3 Features

While spanning trees are familiar from non-projective dependency parsing, features based on the linear order of the words or on lexical identi-

[4] See Georgiadis (2003) for a detailed algorithmic proof, and McDonald et al. (2005) for an illustrative example. Also, we constrain the Chu-Liu-Edmonds MST algorithm to output only *single-root* MSTs, where the (dummy) root has exactly one child (Koo et al., 2007), because multi-root spanning 'forests' are not applicable to our task.
Also, note that we currently assume one node per term. We are following the task description from previous work where the goal is to create a taxonomy for a specific domain (e.g., animals). Within a specific domain, terms typically just have a single sense. However, our algorithms could certainly be adapted to the case of multiple term senses (by treating the different senses as unique nodes in the tree) in future work.

[5] The MST that is found using these edge scores is actually the minimum Bayes risk tree (Goodman, 1996) for an edge accuracy loss function (Smith and Eisner, 2008).

ties or syntactic word classes, which are primary drivers for dependency parsing, are mostly uninformative for taxonomy induction. Instead, inducing taxonomies requires world knowledge to capture the semantic relations between various unseen terms. For this, we use semantic cues to hypernymy and siblinghood via features on simple surface patterns and statistics in large text corpora. We fire features on both the edge and the sibling factors. We first describe all the edge features in detail (Section 3.1 and Section 3.2), and then briefly describe the sibling features (Section 3.3), which are quite similar to the edge ones.

For each edge factor E_{ij}, which represents the potential parent-child term pair (x_i, x_j), we add the surface and semantic features discussed below. Note that since edges are directed, we have separate features for the factors E_{ij} versus E_{ji}.

3.1 Surface Features

Capitalization: Checks which of x_i and x_j are capitalized, with one feature for each value of the tuple (isCap(x_i), isCap(x_j)). The intuition is that leaves of a taxonomy are often proper names and hence capitalized, e.g., (*bison, American bison*). Therefore, the feature for (*true, false*) (i.e., parent capitalized but not the child) gets a substantially negative weight.

Ends with: Checks if x_j ends with x_i, or not. This captures pairs such as (*fish, bony fish*) in our data.

Contains: Checks if x_j contains x_i, or not. This captures pairs such as (*bird, bird of prey*).

Suffix match: Checks whether the k-length suffixes of x_i and x_j match, or not, for $k = 1, 2, \ldots, 7$.

LCS: We compute the longest common substring of x_i and x_j, and create indicator features for rounded-off and binned values of $|LCS|/((|x_i| + |x_j|)/2)$.

Length difference: We compute the signed length difference between x_j and x_i, and create indicator features for rounded-off and binned values of $(|x_j| - |x_i|)/((|x_i| + |x_j|)/2)$. Yang and Callan (2009) use a similar feature.

3.2 Semantic Features

3.2.1 Web n-gram Features

Patterns and counts: Hypernymy for a term pair (*P*=x_i, *C*=x_j) is often signaled by the presence of surface patterns like *C is a P*, *P such as C* in large text corpora, an observation going back to Hearst (1992). For each potential parent-child edge (*P*=x_i, *C*=x_j), we mine the top k strings (based on count) in which both x_i and x_j occur (we use k=200). We collect patterns in both directions, which allows us to judge the correct direction of an edge (e.g., *C is a P* is a positive signal for hypernymy whereas *P is a C* is a negative signal).[6] Next, for each pattern in this top-k list, we compute its normalized pattern count c, and fire an indicator feature on the tuple $(pattern, t)$, for all thresholds t (in a fixed set) s.t. $c \geq t$. Our supervised model then automatically learns which patterns are good indicators of hypernymy.

Pattern order: We add features on the order (direction) in which the pair (x_i, x_j) found a pattern (in its top-k list) – indicator features for boolean values of the four cases: $P \ldots C$, $C \ldots P$, neither direction, and both directions. Ritter et al. (2009) used the 'both' case of this feature.

Individual counts: We also compute the individual Web-scale term counts c_{x_i} and c_{x_j}, and add a comparison feature ($c_{x_i} > c_{x_j}$), plus features on values of the signed count difference $(|c_{x_i}| - |c_{x_j}|)/((|c_{x_i}| + |c_{x_j}|)/2)$, after rounding off, and binning at multiple granularities. The intuition is that this feature could learn whether the relative popularity of the terms signals their hypernymy direction.

3.2.2 Wikipedia Abstract Features

The Web n-grams corpus has broad coverage but is limited to up to 5-grams, so it may not contain pattern-based evidence for various longer multiword terms and pairs. Therefore, we supplement it with a full-sentence resource, namely *Wikipedia abstracts*, which are concise descriptions (hence useful to signal hypernymy) of a large variety of world entities.

Presence and distance: For each potential edge (x_i, x_j), we mine patterns from all abstracts in which the two terms co-occur in either order, allowing a maximum term distance of 20 (because beyond that, co-occurrence may not imply a relation). We add a presence feature based on whether the process above found at least one pattern for that term pair, or not. We also fire features on the value of the minimum distance d_{min} at which

[6] We also allow patterns with surrounding words, e.g., ***the*** *C is a P* and *C , P* ***of***.

the two terms were found in some abstract (plus thresholded versions).

Patterns: For each term pair, we take the top-k' patterns (based on count) of length up to l from its full list of patterns, and add an indicator feature on each pattern string (without the counts). We use k'=5, l=10. Similar to the Web n-grams case, we also fire Wikipedia-based pattern order features.

3.3 Sibling Features

We also incorporate similar features on sibling factors. For each sibling factor S_{ijk} which represents the potential parent-children term triple (x_i, x_j, x_k), we consider the potential sibling term pair (x_j, x_k). Siblinghood for this pair would be indicated by the presence of surface patterns such as *either* C_1 *or* C_2, C_1 *is similar to* C_2 in large corpora. Hence, we fire Web n-gram pattern features and Wikipedia presence, distance, and pattern features, similar to those described above, on each potential sibling term pair.[7] The main difference here from the edge factors is that the sibling factors are symmetric (in the sense that S_{ijk} is redundant to S_{ikj}) and hence the patterns are undirected. Therefore, for each term pair, we first symmetrize the collected Web n-grams and Wikipedia patterns by accumulating the counts of symmetric patterns like *rats or squirrels* and *squirrels or rats*.[8]

4 Related Work

In our work, we assume a known term set and do not address the problem of extracting related terms from text. However, a great deal of past work has considered automating this process, typically taking one of two major approaches. The clustering-based approach (Lin, 1998; Lin and Pantel, 2002; Davidov and Rappoport, 2006; Yamada et al., 2009) discovers relations based on the assumption that similar concepts appear in similar contexts (Harris, 1954). The pattern-based approach uses special lexico-syntactic patterns to extract pairwise relation lists (Phillips and Riloff, 2002; Girju et al., 2003; Pantel and Pennacchiotti, 2006; Suchanek et al., 2007; Ritter et al., 2009; Hovy et al., 2009; Baroni et al., 2010; Ponzetto and Strube, 2011) and semantic classes or class-instance pairs (Riloff and Shepherd, 1997; Katz and Lin, 2003; Paşca, 2004; Etzioni et al., 2005; Talukdar et al., 2008).

We focus on the second step of taxonomy induction, namely the structured organization of terms into a complete and coherent tree-like hierarchy.[9] Early work on this task assumes a starting partial taxonomy and inserts missing terms into it. Widdows (2003) place unknown words into a region with the most semantically-similar neighbors. Snow et al. (2006) add novel terms by greedily maximizing the conditional probability of a set of relational evidence given a taxonomy. Yang and Callan (2009) incrementally cluster terms based on a pairwise semantic distance. Lao et al. (2012) extend a knowledge base using a random walk model to learn binary relational inference rules.

However, the task of inducing full taxonomies without assuming a substantial initial partial taxonomy is relatively less well studied. There is some prior work on the related task of hierarchical clustering, or grouping together of semantically related words (Cimiano et al., 2005; Cimiano and Staab, 2005; Poon and Domingos, 2010; Fountain and Lapata, 2012). The task we focus on, though, is the discovery of direct taxonomic relationships (e.g., hypernymy) between words.

We know of two closely-related previous systems, Kozareva and Hovy (2010) and Navigli et al. (2011), that build full taxonomies from scratch. Both of these systems use a process that starts by finding basic level terms (leaves of the final taxonomy tree, typically) and then using relational patterns (hand-selected ones in the case of Kozareva and Hovy (2010), and ones learned separately by a pairwise classifier on manually annotated co-occurrence patterns for Navigli and Velardi (2010), Navigli et al. (2011)) to find intermediate terms and all the attested hypernymy links between them.[10] To prune down the resulting tax-

[7] One can also add features on the full triple (x_i, x_j, x_k) but most such features will be sparse.

[8] All the patterns and counts for our Web and Wikipedia edge and sibling features described above are extracted after stemming the words in the terms, the n-grams, and the abstracts (using the Porter stemmer). Also, we threshold the features (to prune away the sparse ones) by considering only those that fire for at least t trees in the training data ($t = 4$ in our experiments).
Note that one could also add various complementary types of useful features presented by previous work, e.g., bootstrapping using syntactic heuristics (Phillips and Riloff, 2002), dependency patterns (Snow et al., 2006), doubly anchored patterns (Kozareva et al., 2008; Hovy et al., 2009), and Web definition classifiers (Navigli et al., 2011).

[9] Determining the set of input terms is orthogonal to our work, and our method can be used in conjunction with various term extraction approaches described above.

[10] Unlike our system, which assumes a complete set of terms and only attempts to induce the taxonomic structure,

onomy graph, Kozareva and Hovy (2010) use a procedure that iteratively retains the longest paths between root and leaf terms, removing conflicting graph edges as they go. The end result is acyclic, though not necessarily a tree; Navigli et al. (2011) instead use the longest path intuition to weight edges in the graph and then find the highest weight taxonomic tree using a standard MST algorithm.

Our work differs from the two systems above in that ours is the first discriminatively trained, structured probabilistic model over the full space of taxonomy trees that uses structured inference via spanning tree algorithms (MST and MTT) through both the learning and decoding phases. Our model also automatically learns relational patterns as a part of the taxonomic training phase, instead of relying on hand-picked rules or pairwise classifiers on manually annotated co-occurrence patterns, and it is the first end-to-end (i.e., non-incremental) system to include heterogeneous relational information via sibling (e.g., coordination) patterns.

5 Experiments

5.1 Data and Experimental Regime

We considered two distinct experimental setups, one that illustrates the general performance of our model by reproducing various medium-sized WordNet domains, and another that facilitates comparison to previous work by reproducing the much larger *animal* subtree provided by Kozareva and Hovy (2010).

General setup: In order to test the accuracy of structured prediction on medium-sized full-domain taxonomies, we extracted from WordNet 3.0 all bottomed-out full subtrees which had a tree-height of 3 (i.e., 4 nodes from root to leaf), and contained (10, 50] terms.[11] This gives us 761 non-overlapping trees, which we partition into 70/15/15% (533/114/114 trees) train/dev/test sets.

Comparison setup: We also compare our method (as closely as possible) with related previous work by testing on the much larger *animal* subtree made available by Kozareva and Hovy (2010), who created this dataset by selecting a set of 'harvested' terms and retrieving all the WordNet hypernyms between each input term and the root (i.e., *animal*), resulting in ∼700 terms and ∼4,300 *is-a* ancestor-child links.[12] Our training set for this *animal* test case was generated from WordNet using the following process: First, we strictly remove the *full* animal subtree from WordNet in order to avoid any possible overlap with the test data. Next, we create random 25-sized trees by picking random nodes as singleton trees, and repeatedly adding child edges from WordNet to the tree. This process gives us a total of ∼1600 training trees.[13]

Feature sources: The n-gram semantic features are extracted from the Google n-grams corpus (Brants and Franz, 2006), a large collection of English n-grams (for $n = 1$ to 5) and their frequencies computed from almost 1 trillion tokens (95 billion sentences) of Web text. The Wikipedia abstracts are obtained via the publicly available dump, which contains almost ∼4.1 million articles.[14] Preprocessing includes standard XML parsing and tokenization. Efficient collection of feature statistics is important because these must be extracted for millions of query pairs (for each potential edge and sibling pair in each term set). For this, we use a hash-trie on term pairs (similar to that of Bansal and Klein (2011)), and scan once through the n-gram (or abstract) set, skipping many n-grams (or abstracts) based on fast checks of missing unigrams, exceeding length, suffix mismatches, etc.

5.2 Evaluation Metric

Ancestor F1: Measures the precision, recall, and $F_1 = 2PR/(P+R)$ of correctly predicted ances-

both these systems include term discovery in the taxonomy building process.

[11] Subtrees that had a smaller or larger tree height were discarded in order to avoid overlap between the training and test divisions. This makes it a much *stricter setting* than other tasks such as parsing, which usually has repeated sentences, clauses and phrases between training and test sets.
To project WordNet synsets to terms, we used the first (most frequent) term in each synset. A few WordNet synsets have multiple parents so we only keep the first of each such pair of overlapping trees. We also discard a few trees with duplicate terms because this is mostly due to the projection of different synsets to the same term, and theoretically makes the tree a graph.

[12] This is somewhat different from our general setup where we work with any given set of terms; they start with a large set of leaves which have substantial Web-based relational information based on their selected, hand-picked patterns. Their data is available at `http://www.isi.edu/~kozareva/downloads.html`.

[13] We tried this training regimen as different from that of the general setup (which contains only bottomed-out subtrees), so as to match the *animal* test tree, which is of depth 12 and has intermediate nodes from higher up in WordNet.

[14] We used the *20130102* dump.

System	P	R	F1
Edges-Only Model			
Baseline	5.9	8.3	6.9
Surface Features	17.5	41.3	24.6
Semantic Features	37.0	49.1	42.2
Surface+Semantic	41.1	54.4	46.8
Edges + Siblings Model			
Surface+Semantic	**53.1**	**56.6**	**54.8**
Surface+Semantic (Test)	48.0	55.2	51.4

Table 1: Main results on our general setup. On the development set, we present incremental results on the edges-only model where we start with the chance baseline, then use surface features only, semantic features only, and both. Finally, we add sibling factors and features to get results for the full, edges+siblings model with all features, and also report the final test result for this setting.

System	P	R	F1
Previous Work			
Kozareva2010	**98.6**	36.2	52.9
Navigli2011**	97.0**	43.7**	60.3**
This Paper			
Fixed Prediction	84.2	**55.1**	**66.6**
Free Prediction	79.3	49.0	60.6

Table 2: Comparison results on the *animal* dataset of Kozareva and Hovy (2010). Here, 'Kozareva2010' refers to Kozareva and Hovy (2010) and 'Navigli2011' refers to Navigli et al. (2011). For appropriate comparison to each previous work, we show our results both for the 'Fixed Prediction' setup, which assumes the true root and leaves, and for the 'Free Prediction' setup, which doesn't assume any prior information. The $\star\star$ results of Navigli et al. (2011) represent a different ground-truth data condition, making them incomparable to our results; see Section 5.3 for details.

tors, i.e., pairwise *is-a* relations:

$$P = \frac{|\text{isa}_{\text{gold}} \cap \text{isa}_{\text{predicted}}|}{|\text{isa}_{\text{predicted}}|}, \; R = \frac{|\text{isa}_{\text{gold}} \cap \text{isa}_{\text{predicted}}|}{|\text{isa}_{\text{gold}}|}$$

5.3 Results

Table 1 shows our main results for ancestor-based evaluation on the general setup. We present a development set ablation study where we start with the edges-only model (Figure 2a) and its random tree baseline (which chooses any arbitrary spanning tree for the term set). Next, we show results on the edges-only model with surface features (Section 3.1), semantic features (Section 3.2), and both. We see that both surface and semantic features make substantial contributions, and they also stack. Finally, we add the sibling factors and features (Figure 2b, Section 3.3), which further improves the results significantly (8% absolute and 15% relative error reduction over the edges-only results on the ancestor F1 metric). The last row shows the final test set results for the full model with all features.

Table 2 shows our results for comparison to the larger *animal* dataset of Kozareva and Hovy (2010).[15] In the table, 'Kozareva2010' refers to Kozareva and Hovy (2010) and 'Navigli2011' refers to Navigli et al. (2011).[16] For appropriate comparison to each previous work, we show results for two different setups. The first setup 'Fixed Prediction' assumes that the model knows the true root and leaves of the taxonomy to provide for a somewhat fairer comparison to Kozareva and Hovy (2010). We get substantial improvements on ancestor-based recall and F1 (a 29% relative error reduction). The second setup 'Free Prediction' assumes no prior knowledge and predicts the full tree (similar to the general setup case). On this setup, we do compare as closely as possible to Navigli et al. (2011) and see a small gain in F1, but regardless, we should note that their results are incomparable (denoted by $\star\star$ in Table 2) because they have a different ground-truth data condition: their definition and hypernym extraction phase involves using the Google `define` keyword, which often returns WordNet glosses itself.

We note that previous work achieves higher ancestor precision, while our approach achieves a more even balance between precision and recall. Of course, precision and recall should both ideally be high, even if some applications weigh one over the other. This is why our tuning optimized for F1, which represents a neutral combination for comparison, but other F_α metrics could also be optimized. In this direction, we also tried an experiment on *precision-based decoding* (for the 'Free Prediction' scenario), where we discard any edges with score (i.e., the belief odds ratio described in Section 2.4) less than a certain threshold. This allowed us to achieve high values of precision (e.g., 90.8%) at still high enough F1 values (e.g., 61.7%).

[15] These results are for the 1st order model due to the scale of the animal taxonomy (∼700 terms). For scaling the 2nd order sibling model, one can use approximations, e.g., pruning the set of sibling factors based on 1st order link marginals, or a hierarchical coarse-to-fine approach based on taxonomy induction on subtrees, or a greedy approach of adding a few sibling factors at a time. This is future work.

[16] The Kozareva and Hovy (2010) ancestor results are obtained by using the output files provided on their webpage.

Hypernymy features	
C and other P	*> P > C*
C , P of	*C is a P*
C , a P	*P , including C*
C or other P	*P (C*
C : a P	*C , american P*
C - like P	*C , the P*
Siblinghood features	
C_1 *and* C_2	C_1, C_2 *(*
C_1 *or* C_2 *of*	C_1 *and / or* C_2
, C_1 *,* C_2 *and*	*either* C_1 *or* C_2
the C_1 */* C_2	*<s> C₁ and C₂ </s>*

Table 3: Examples of high-weighted hypernymy and siblinghood features learned during development.

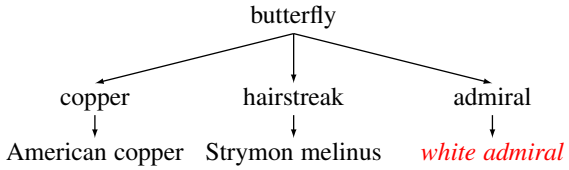


Figure 3: Excerpt from the predicted *butterfly* tree. The terms attached erroneously according to WordNet are marked in red and italicized.

6 Analysis

Table 3 shows some of the hypernymy and siblinghood features given highest weight by our model (in general-setup development experiments). The training process not only rediscovers most of the standard Hearst-style hypernymy patterns (e.g., *C and other P*, *C is a P*), but also finds various novel, intuitive patterns. For example, the pattern *C, american P* is prominent because it captures pairs like *Lemmon, american actor* and *Bryon, american politician*, etc. Another pattern *> P > C* captures webpage navigation breadcrumb trails (representing category hierarchies). Similarly, the algorithm also discovers useful siblinghood features, e.g., *either* C_1 *or* C_2, C_1 *and / or* C_2, etc.

Finally, we look at some specific output errors to give as concrete a sense as possible of some system confusions, though of course any hand-chosen examples must be taken as illustrative. In Figure 3, we attach *white admiral* to *admiral*, whereas the gold standard makes these two terms siblings. In reality, however, white admirals are indeed a species of admirals, so WordNet's ground truth turns out to be incomplete. Another such example is that we place *logistic assessment* in the *evaluation* subtree of *judgment*, but WordNet makes it a direct child of *judgment*. However, other dictionaries do consider logistic assessments to be evaluations. Hence, this illustrates that there may be more than one right answer, and that the low results on this task should only be interpreted as such. In Figure 4, our algorithm did not recognize that *thermos* is a hyponym of *vacuum flask*, and that *jeroboam* is a kind of wine bottle. Here, our Web n-grams dataset (which only contains frequent n-grams) and Wikipedia abstracts do not suffice and we would need to add richer Web data for such world knowledge to be reflected in the features.

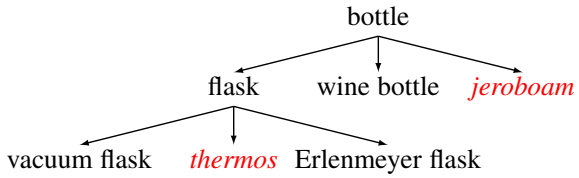


Figure 4: Excerpt from the predicted *bottle* tree. The terms attached erroneously according to WordNet are marked in red and italicized.

7 Conclusion

Our approach to taxonomy induction allows heterogeneous information sources to be combined and balanced in an error-driven way. Direct indicators of hypernymy, such as Hearst-style context patterns, are the core feature for the model and are discovered automatically via discriminative training. However, other indicators, such as coordination cues, can indicate that two words might be siblings, independently of what their shared parent might be. Adding second-order factors to our model allows these two kinds of evidence to be weighed and balanced in a discriminative, structured probabilistic framework. Empirically, we see substantial gains (in ancestor F1) from sibling features, and also over comparable previous work. We also present results on the precision and recall trade-offs inherent in this task.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments. This work was supported by BBN under DARPA contract HR0011-12-C-0014, 973 Program China Grants 2011CBA00300, 2011CBA00301, and NSFC Grants 61033001, 61361136003.

References

Mohit Bansal and Dan Klein. 2011. Web-scale features for full-scale parsing. In *Proceedings of ACL*.

Marco Baroni, Brian Murphy, Eduard Barbu, and Massimo Poesio. 2010. Strudel: A corpus-based semantic model based on properties and types. *Cognitive Science*, 34(2):222–254.

Thorsten Brants and Alex Franz. 2006. The Google Web 1T 5-gram corpus version 1.1. *LDC2006T13*.

Yoeng-Jin Chu and Tseng-Hong Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14(1396-1400):270.

Philipp Cimiano and Steffen Staab. 2005. Learning concept hierarchies from text with a guided agglomerative clustering algorithm. In *Proceedings of the ICML 2005 Workshop on Learning and Extending Lexical Ontologies with Machine Learning Methods*.

Philipp Cimiano, Andreas Hotho, and Steffen Staab. 2005. Learning concept hierarchies from text corpora using formal concept analysis. *Journal of Artificial Intelligence Research*, 24(1):305–339.

Dmitry Davidov and Ari Rappoport. 2006. Efficient unsupervised discovery of word categories using symmetric patterns and high frequency words. In *Proceedings of COLING-ACL*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards B*, 71:233–240.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the Web: An experimental study. *Artificial Intelligence*, 165(1):91–134.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. 2010. Building watson: An overview of the DeepQA project. *AI magazine*, 31(3):59–79.

Trevor Fountain and Mirella Lapata. 2012. Taxonomy induction using hierarchical random graphs. In *Proceedings of NAACL*.

Leonidas Georgiadis. 2003. Arborescence optimization problems solvable by edmonds algorithm. *Theoretical Computer Science*, 301(1):427–437.

Roxana Girju, Adriana Badulescu, and Dan Moldovan. 2003. Learning semantic constraints for the automatic discovery of part-whole relations. In *Proceedings of NAACL*.

Joshua Goodman. 1996. Parsing algorithms and metrics. In *Proceedings of ACL*.

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

Marti Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of COLING*.

Eduard Hovy, Zornitsa Kozareva, and Ellen Riloff. 2009. Toward completeness in concept extraction and classification. In *Proceedings of EMNLP*.

Boris Katz and Jimmy Lin. 2003. Selectively using relations to improve precision in question answering. In *Proceedings of the Workshop on NLP for Question Answering (EACL 2003)*.

Terry Koo, Amir Globerson, Xavier Carreras, and Michael Collins. 2007. Structured prediction models via the matrix-tree theorem. In *Proceedings of EMNLP-CoNLL*.

Zornitsa Kozareva and Eduard Hovy. 2010. A semi-supervised method to learn and construct taxonomies using the Web. In *Proceedings of EMNLP*.

Zornitsa Kozareva, Ellen Riloff, and Eduard Hovy. 2008. Semantic class learning from the web with hyponym pattern linkage graphs. In *Proceedings of ACL*.

Ni Lao, Amarnag Subramanya, Fernando Pereira, and William W. Cohen. 2012. Reading the web with learned syntactic-semantic inference rules. In *Proceedings of EMNLP*.

Dekang Lin and Patrick Pantel. 2002. Concept discovery from text. In *Proceedings of COLING*.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of COLING*.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT-EMNLP*.

Roberto Navigli and Paola Velardi. 2010. Learning word-class lattices for definition and hypernym extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.

Roberto Navigli, Paola Velardi, and Stefano Faralli. 2011. A graph-based algorithm for inducing lexical taxonomies from scratch. In *Proceedings of IJCAI*.

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of COLING-ACL*.

Marius Paşca. 2004. Acquisition of categorized named entities for web search. In *Proceedings of CIKM*.

Marco Pennacchiotti and Patrick Pantel. 2006. Ontologizing semantic relations. In *Proceedings of COLING-ACL*.

William Phillips and Ellen Riloff. 2002. Exploiting strong syntactic heuristics and co-training to learn semantic lexicons. In *Proceedings of EMNLP*.

Simone Paolo Ponzetto and Michael Strube. 2011. Taxonomy induction based on a collaboratively built knowledge repository. *Artificial Intelligence*, 175(9):1737–1756.

Hoifung Poon and Pedro Domingos. 2010. Unsupervised ontology induction from text. In *Proceedings of ACL*.

Ellen Riloff and Jessica Shepherd. 1997. A corpus-based approach for building semantic lexicons. In *Proceedings of EMNLP*.

Alan Ritter, Stephen Soderland, and Oren Etzioni. 2009. What is this, anyway: Automatic hypernym discovery. In *Proceedings of AAAI Spring Symposium on Learning by Reading and Learning to Read*.

David A. Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *Proceedings of EMNLP*.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of COLING-ACL*.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of WWW*.

Partha Pratim Talukdar, Joseph Reisinger, Marius Paşca, Deepak Ravichandran, Rahul Bhagat, and Fernando Pereira. 2008. Weakly-supervised acquisition of labeled class instances using graph random walks. In *Proceedings of EMNLP*.

Robert E. Tarjan. 1977. Finding optimum branchings. *Networks*, 7:25–35.

William T. Tutte. 1984. Graph theory. *Addison-Wesley*.

Dominic Widdows. 2003. Unsupervised methods for developing taxonomies by combining syntactic and statistical information. In *Proceedings of HLT-NAACL*.

Ichiro Yamada, Kentaro Torisawa, Jun'ichi Kazama, Kow Kuroda, Masaki Murata, Stijn De Saeger, Francis Bond, and Asuka Sumida. 2009. Hypernym discovery based on distributional similarity and hierarchical structures. In *Proceedings of EMNLP*.

Hui Yang and Jamie Callan. 2009. A metric-based framework for automatic taxonomy induction. In *Proceedings of ACL-IJCNLP*.

A Provably Correct Learning Algorithm for Latent-Variable PCFGs

Shay B. Cohen
School of Informatics
University of Edinburgh
scohen@inf.ed.ac.uk

Michael Collins
Department of Computer Science
Columbia University
mcollins@cs.columbia.edu

Abstract

We introduce a provably correct learning algorithm for latent-variable PCFGs. The algorithm relies on two steps: first, the use of a matrix-decomposition algorithm applied to a co-occurrence matrix estimated from the parse trees in a training sample; second, the use of EM applied to a convex objective derived from the training samples in combination with the output from the matrix decomposition. Experiments on parsing and a language modeling problem show that the algorithm is efficient and effective in practice.

1 Introduction

Latent-variable PCFGs (L-PCFGs) (Matsuzaki et al., 2005; Petrov et al., 2006) give state-of-the-art performance on parsing problems. The standard approach to parameter estimation in L-PCFGs is the EM algorithm (Dempster et al., 1977), which has the usual problems with local optima. Recent work (Cohen et al., 2012) has introduced an alternative algorithm, based on spectral methods, which has provable guarantees. Unfortunately this algorithm does not return parameter estimates for the underlying L-PCFG, instead returning the parameter values up to an (unknown) linear transform. In practice, this is a limitation.

We describe an algorithm that, like EM, returns estimates of the original parameters of an L-PCFG, but, unlike EM, does not suffer from problems of local optima. The algorithm relies on two key ideas:

1) A *matrix decomposition algorithm* (section 5) which is applicable to matrices Q of the form $Q_{f,g} = \sum_h p(h)p(f \mid h)p(g \mid h)$ where $p(h), p(f \mid h)$ and $p(g \mid h)$ are multinomial distributions. This matrix form has clear relevance to latent variable models. We apply the matrix decomposition algorithm to a co-occurrence matrix that can be estimated directly from a training set consisting of parse trees without latent annotations. The resulting parameter estimates give us significant leverage over the learning problem.

2) *Optimization of a convex objective function using EM.* We show that once the matrix decomposition step has been applied, parameter estimation of the L-PCFG can be reduced to a convex optimization problem that is easily solved by EM.

The algorithm provably learns the parameters of an L-PCFG (theorem 1), under an assumption that each latent state has at least one "pivot" feature. This assumption is similar to the "pivot word" assumption used by Arora et al. (2013) and Arora et al. (2012) in the context of learning topic models.

We describe experiments on learning of L-PCFGs, and also on learning of the latent-variable language model of Saul and Pereira (1997). A hybrid method, which uses our algorithm as an initializer for EM, performs at the same accuracy as EM, but requires significantly fewer iterations for convergence: for example in our L-PCFG experiments, it typically requires 2 EM iterations for convergence, as opposed to 20-40 EM iterations for initializers used in previous work.

While this paper's focus is on L-PCFGs, the techniques we describe are likely to be applicable to many other latent-variable models used in NLP.

2 Related Work

Recently a number of researchers have developed provably correct algorithms for parameter estimation in latent variable models such as hidden Markov models, topic models, directed graphical models with latent variables, and so on (Hsu et al., 2009; Bailly et al., 2010; Siddiqi et al., 2010; Parikh et al., 2011; Balle et al., 2011; Arora et al., 2013; Dhillon et al., 2012; Anandkumar et al., 2012; Arora et al., 2012; Arora et al., 2013). Many of these algorithms have their roots in spectral methods such as canonical correlation analysis (CCA) (Hotelling, 1936), or higher-order tensor decompositions. Previous work (Cohen et al., 2012; Cohen et al., 2013) has developed a spectral method for learning of L-PCFGs; this method learns parameters of the model up to an unknown

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1052–1061,
Baltimore, Maryland, USA, June 23-25 2014.

linear transformation, which cancels in the inside-outside calculations for marginalization over latent states in the L-PCFG. The lack of direct parameter estimates from this method leads to problems with negative or unnormalized probablities; the method does not give parameters that are interpretable, or that can be used in conjunction with other algorithms, for example as an initializer for EM steps that refine the model.

Our work is most directly related to the algorithm for parameter estimation in topic models described by Arora et al. (2013). This algorithm forms the core of the matrix decomposition algorithm described in section 5.

3 Background

This section gives definitions and notation for L-PCFGs, taken from (Cohen et al., 2012).

3.1 L-PCFGs: Basic Definitions

An L-PCFG is an 8-tuple $(\mathcal{N}, \mathcal{I}, \mathcal{P}, m, n, \pi, t, q)$ where: $\mathcal{N}$ is the set of non-terminal symbols in the grammar. $\mathcal{I} \subset \mathcal{N}$ is a finite set of *in-terminals*. $\mathcal{P} \subset \mathcal{N}$ is a finite set of *pre-terminals*. We assume that $\mathcal{N} = \mathcal{I} \cup \mathcal{P}$, and $\mathcal{I} \cap \mathcal{P} = \emptyset$. Hence we have partitioned the set of non-terminals into two subsets. $[m]$ is the set of possible hidden states.[1] $[n]$ is the set of possible words. For all $(a, b, c) \in \mathcal{I} \times \mathcal{N} \times \mathcal{N}$, and $(h_1, h_2, h_3) \in [m] \times [m] \times [m]$, we have a context-free rule $a(h_1) \rightarrow b(h_2) \;\; c(h_3)$. The rule has an associated parameter $t(a \rightarrow b\, c, h_2, h_3 \mid a, h_1)$. For all $a \in \mathcal{P}$, $h \in [m]$, $x \in [n]$, we have a context-free rule $a(h) \rightarrow x$. The rule has an associated parameter $q(a \rightarrow x \mid a, h)$. For all $a \in \mathcal{I}$, $h \in [m]$, $\pi(a, h)$ is a parameter specifying the probability of $a(h)$ being at the root of a tree.

A *skeletal tree* (s-tree) is a sequence of rules $r_1 \ldots r_N$ where each r_i is either of the form $a \rightarrow b \;\; c$ or $a \rightarrow x$. The rule sequence forms a top-down, left-most derivation under a CFG with skeletal rules.

A *full tree* consists of an s-tree $r_1 \ldots r_N$, together with values $h_1 \ldots h_N$. Each h_i is the value for the hidden variable for the left-hand-side of rule r_i. Each h_i can take any value in $[m]$.

For a given skeletal tree $r_1 \ldots r_N$, define a_i to be the non-terminal on the left-hand-side of rule r_i. For any $i \in [N]$ such that r_i is of the form $a \rightarrow b\, c$, define $h_i^{(2)}$ and $h_i^{(3)}$ as the hidden state value of the left and right child respectively. The model then defines a distribution as

$$p(r_1 \ldots r_N, h_1 \ldots h_N) = \pi(a_1, h_1) \prod_{i: a_i \in \mathcal{I}} t(r_i, h_i^{(2)}, h_i^{(3)} \mid a_i, h_i) \prod_{i: a_i \in \mathcal{P}} q(r_i \mid a_i, h_i)$$

The distribution over skeletal trees is $p(r_1 \ldots r_N) = \sum_{h_1 \ldots h_N} p(r_1 \ldots r_N, h_1 \ldots h_N)$.

3.2 Definition of Random Variables

Throughout this paper we will make reference to random variables derived from the distribution over full trees from an L-PCFG. These random variables are defined as follows. First, we select a random internal node, from a random tree, as follows: 1) Sample a full tree $r_1 \ldots r_N, h_1 \ldots h_N$ from the PMF $p(r_1 \ldots r_N, h_1 \ldots h_N)$; 2) Choose a node i uniformly at random from $[N]$. We then give the following definition:

Definition 1 (Random Variables)**.** If the rule r_i for the node i is of the form $a \rightarrow b\, c$, we define random variables as follows: R_1 is equal to the rule r_i (e.g., `NP` $\rightarrow$ `D N`). A, B, C are the labels for node i, the left child of node i, and the right child of node i respectively. (E.g., $A =$ `NP`, $B =$ `D`, $C =$ `N`.) T_1 is the inside tree rooted at node i. T_2 is the inside tree rooted at the left child of node i, and T_3 is the inside tree rooted at the right child of node i. O is the outside tree at node i. H_1, H_2, H_3 are the hidden variables associated with node i, the left child of node i, and the right child of node i respectively. E is equal to 1 if node i is at the root of the tree (i.e., $i = 1$), 0 otherwise.

If the rule r_i for the selected node i is of the form $a \rightarrow x$, we have random variables R_1, T_1, H_1, A_1, O, E as defined above, but H_2, H_3, T_2, T_3, B, and C are not defined.

4 The Learning Algorithm for L-PCFGs

Our goal is to design a learning algorithm for L-PCFGs. The input to the algorithm will be a training set consisting of skeletal trees, assumed to be sampled from some underlying L-PCFG. The output of the algorithm will be estimates for the π, t, and q parameters. The training set does not include values for the latent variables; this is the main challenge in learning.

This section focuses on an algorithm for recovery of the t parameters. A description of the algorithms for recovery of the π and q parameters is deferred until section 6.1 of this paper; these

[1] For any integer n, we use $[n]$ to denote the set $\{1, 2, \ldots n\}$.

steps are straightforward once we have derived the method for the t parameters.

We describe an algorithm that correctly recovers the parameters of an L-PCFG as the size of the training set goes to infinity (this statement is made more precise in section 4.2). The algorithm relies on an assumption—the "pivot" assumption—that we now describe.

4.1 Features, and the Pivot Assumption

We assume a function τ from inside trees to a finite set $\mathcal{F}$, and a function ρ that maps outside trees to a finite set $\mathcal{G}$. The function $\tau(t)$ ($\rho(o)$) can be thought of as a function that maps an inside tree t (outside tree o) to an underlying feature. As one example, the function $\tau(t)$ might return the context-free rule at the root of the inside tree t; in this case the set $\mathcal{F}$ would be equal to the set of all context-free rules in the grammar. As another example, the function $\rho(o)$ might return the context-free rule at the foot of the outside tree o.

In the more general case, we might have K separate functions $\tau^{(k)}(t)$ for $k = 1 \ldots K$ mapping inside trees to K separate features, and similarly we might have multiple features for outside trees. Cohen et al. (2013) describe one such feature definition, where features track single context-free rules as well as larger fragments such as two or three-level sub-trees. For simplicity of presentation we describe the case of single features $\tau(t)$ and $\rho(o)$ for the majority of this paper. The extension to multiple features is straightforward, and is discussed in section 6.2; the flexibility allowed by multiple features is important, and we use multiple features in our experiments.

Given functions τ and ρ, we define additional random variables: $F = \tau(T_1), F_2 = \tau(T_2), F_3 = \tau(T_3)$, and $G = \rho(O)$.

We can now give the following assumption:

Assumption 1 (The Pivot Assumption)**.** *Under the L-PCFG being learned, there exist values $\alpha > 0$ and $\beta > 0$ such that for each non-terminal a, for each hidden state $h \in [m]$, the following statements are true: 1) $\exists f \in \mathcal{F}$ such that $P(F = f \mid H_1 = h, A = a) > \alpha$ and for all $h' \neq h$, $P(F = f \mid H_1 = h', A = a) = 0$; 2) $\exists g \in \mathcal{G}$ such that $P(G = g \mid H_1 = h, A = a) > \beta$ and for all $h' \neq h$, $P(G = g \mid H_1 = h', A = a) = 0$.*

This assumption is very similar to the assumption made by Arora et al. (2012) in the context of learning topic models. It implies that for each (a, h) pair, there are inside and outside tree features—which following Arora et al. (2012) we refer to as pivot features—that occur only[2] in the presence of latent-state value h. As in (Arora et al., 2012), the pivot features will give us considerable leverage in learning of the model.

4.2 The Learning Algorithm

Figure 1 shows the learning algorithm for L-PCFGs. The algorithm consists of the following steps:

Step 0: Calculate estimates $\hat{p}(a \rightarrow b\ c \mid a)$, $\hat{p}(g, f_2, f_3 \mid a \rightarrow b\ c)$ and $\hat{p}(f, g \mid a)$. These estimates are easily calculated using counts taken from the training examples.

Step 1: Calculate values $\hat{r}(f \mid h, a)$ and $\hat{s}(g \mid h, a)$; these are estimates of $p(f \mid h_1, a)$ and $p(g \mid h_1, a)$ respectively. This step is achieved using a matrix decomposition algorithm, described in section 5 of this paper, on the matrix $\hat{Q}^a$ with entries $[\hat{Q}^a]_{f,g} = \hat{p}(f, g \mid a)$.

Step 2: Use the EM algorithm to find $\hat{t}$ values that maximize the objective function in Eq. 1 (see figure 1). Crucially, this is a convex optimization problem, and the EM algorithm will converge to the global maximum of this likelihood function.

Step 3: Rule estimates are calculated using an application of the laws of probability.

Before giving a theorem concerning correctness of the algorithm we introduce two assumptions:

Assumption 2 (Strict Convexity)**.** *If we have the equalities $\hat{s}(g \mid h_1, a) = P(G = g \mid H_1 = h_1, A = a)$, $\hat{r}(f_2 \mid h_2, b) = P(F_2 = f_2 \mid H_2 = h_2, B = b)$ and $\hat{r}(f_3 \mid h_3, c) = P(F_3 = f_3 \mid H_2 = h_3, C = c)$, then the function in Eq. 1 (figure 1) is strictly concave.*

The function in Eq. 1 is always concave; this assumption adds the restriction that the function must be *strictly* concave—that is, it has a unique global maximum—in the case that the $\hat{r}$ and $\hat{s}$ estimates are exact estimates.

Assumption 3 (Infinite Data)**.** *After running Step 0 of the algorithm we have*

$$\begin{aligned} \hat{p}(a \rightarrow b\ c \mid a) &= p(a \rightarrow b\ c \mid a) \\ \hat{p}(g, f_2, f_3 \mid a \rightarrow b\ c) &= p(g, f_2, f_3 \mid a \rightarrow b\ c) \\ \hat{p}(f, g \mid a) &= p(f, g \mid a) \end{aligned}$$

where $p(\ldots)$ is the probability under the underlying L-PCFG.

[2] The requirements $P(F = f \mid H_1 = h', A = a) = 0$ and $P(G = g \mid H_1 = h', A = a) = 0$ are almost certainly overly strict; in theory and practice these probabilities should be able to take small but strictly positive values.

We use the term "infinite data" because under standard arguments, $\hat{p}(\ldots)$ converges to $p(\ldots)$ as M goes to ∞.

The theorem is then as follows:

Theorem 1. *Consider the algorithm in figure 1. Assume that assumptions 1-3 (the pivot, strong convexity, and infinite data assumptions) hold for the underlying L-PCFG. Then there is some permutation $\sigma : [m] \rightarrow [m]$ such that for all $a \rightarrow b\ c, h_1, h_2, h_3$,*

$$\begin{aligned} & \hat{t}(a \rightarrow b\ c, h_2, h_3 \mid a \rightarrow b\ c, h_1) \\ = \ & t(a \rightarrow b\ c, \sigma(h_2), \sigma(h_3) \mid a \rightarrow b\ c, \sigma(h_1)) \end{aligned}$$

where $\hat{t}$ are the parameters in the output, and t are the parameters of the underlying L-PCFG.

This theorem states that under assumptions 1-3, the algorithm correctly learns the t parameters of an L-PCFG, up to a permutation over the latent states defined by σ. Given the assumptions we have made, it is not possible to do better than recovering the correct parameter values up to a permutation, due to symmetries in the model. Assuming that the π and q parameters are recovered in addition to the t parameters (see section 6.1), the resulting model will define exactly the same distribution over full trees as the underlying L-PCFG up to this permutation, and will define exactly the same distribution over skeletal trees, so in this sense the permutation is benign.

Proof of theorem 1: Under the assumptions of the theorem, $\hat{Q}^a_{f,g} = p(f, g \mid a) = \sum_h p(h \mid a) p(f \mid h, a) p(g \mid h, a)$. Under the pivot assumption, and theorem 2 of section 5, step 1 (the matrix decomposition step) will therefore recover values $\hat{r}$ and $\hat{s}$ such that $\hat{r}(f \mid h, a) = p(f \mid \sigma(h), a)$ and $\hat{s}(g \mid h, a) = p(g \mid \sigma(h), a)$ for some permutation $\sigma : [m] \rightarrow [m]$. For simplicity, assume that $\sigma(j) = j$ for all $j \in [m]$ (the argument for other permutations involves a straightforward extension of the following argument). Under the assumptions of the theorem, $\hat{p}(g, f_2, f_3 \mid a \rightarrow b\ c) = p(g, f_2, f_3 \mid a \rightarrow b\ c)$, hence the function being optimized in Eq. 1 is equal to

$$\sum_{g, f_2, f_3} p(g, f_2, f_3 \mid a \rightarrow b\ c) \log \kappa(g, f_2, f_3)$$

where

$$\begin{aligned} \kappa(g, f_2, f_3) = \sum_{h_1, h_2, h_3} & \big(\hat{t}(h_1, h_2, h_3 \mid a \rightarrow b\ c) \\ & \times p(g \mid h_1, a) p(f_2 \mid h_2, b) p(f_3 \mid h_3, c)\big) \end{aligned}$$

Now consider the optimization problem in Eq. 1. By standard results for cross entropy, the maximum of the function

$$\sum_{g, f_2, f_3} p(g, f_2, f_3 \mid a \rightarrow b\ c) \log q(g, f_2, f_3 \mid a \rightarrow b\ c)$$

with respect to the q values is achieved at $q(g, f_2, f_3 \mid a \rightarrow b\ c) = p(g, f_2, f_3 \mid a \rightarrow b\ c)$. In addition, under the assumptions of the L-PCFG,

$$\begin{aligned} & p(g, f_2, f_3 \mid a \rightarrow b\ c) \\ = & \sum_{h_1, h_2, h_3} (p(h_1, h_2, h_3 \mid a \rightarrow b\ c) \\ & \times p(g \mid h_1, a) p(f_2 \mid h_2, b) p(f_3 \mid h_3, c)) \end{aligned}$$

Hence the maximum of Eq. 1 is achieved at

$$\hat{t}(h_1, h_2, h_3 \mid a \rightarrow b\ c) = p(h_1, h_2, h_3 \mid a \rightarrow b\ c) \tag{2}$$

because this gives $\kappa(g, f_2, f_3) = p(g, f_2, f_3 \mid a \rightarrow b\ c)$. Under the strict convexity assumption the maximum of Eq. 1 is unique, hence the $\hat{t}$ values must satisfy Eq. 2. Finally, it follows from Eq. 2, and the equality $\hat{p}(a \rightarrow b\ c \mid a) = p(a \rightarrow b\ c \mid a)$, that Step 3 of the algorithm gives $\hat{t}(a \rightarrow b\ c, h_2, h_3 \mid a, h_1) = t(a \rightarrow b\ c, h_2, h_3 \mid a, h_1)$. □

We can now see how the strict convexity assumption is needed. Without this assumption, there may be multiple settings for $\hat{t}$ that achieve $\kappa(g, f_2, f_3) = p(g, f_2, f_3 \mid a \rightarrow b\ c)$; the values $\hat{t}(h_1, h_2, h_3 \mid a \rightarrow b\ c) = p(h_1, h_2, h_3 \mid a \rightarrow b\ c)$ will be included in this set of solutions, but other, inconsistent solutions will also be included.

As an extreme example of the failure of the strict convexity assumption, consider a feature-vector definition with $|\mathcal{F}| = |\mathcal{G}| = 1$. In this case the function in Eq. 1 reduces to $\log \sum_{h_1, h_2, h_3} \hat{t}(h_1, h_2, h_3 \mid a \rightarrow b\ c)$. This function has a maximum value of 0, achieved at all values of $\hat{t}$. Intuitively, this definition of inside and outside tree features loses all information about the latent states, and does not allow successful learning of the underlying L-PCFG. More generally, it is clear that the strict convexity assumption will depend directly on the choice of feature functions $\tau(t)$ and $\rho(o)$.

Remark: The infinite data assumption, and sample complexity. The infinite data assumption deserves more discussion. It is clearly a strong assumption that there is sufficient data for

Input: A set of M skeletal trees sampled from some underlying L-PCFG. The count[...] function counts the number of times that event ... occurs in the training sample. For example, count$[A = a]$ is the number of times random variable A takes value a in the training sample.

Step 0: Calculate the following estimates from the training samples:

- $\hat{p}(a \rightarrow b\ c \mid a) = \text{count}[R_1 = a \rightarrow b\ c]/\text{count}[A = a]$
- $\hat{p}(g, f_2, f_3 \mid a \rightarrow b\ c) = \text{count}[G = g, F_2 = f_2, F_3 = f_3, R_1 = a \rightarrow b\ c]/\text{count}[R_1 = a \rightarrow b\ c]$
- $\hat{p}(f, g \mid a) = \text{count}[F = f, G = g, A = a]/\text{count}[A = a]$
- $\forall a \in \mathcal{I}$, define a matrix $\hat{Q}^a \in \mathbb{R}^{d \times d'}$ where $d = |\mathcal{F}|$ and $d' = |\mathcal{G}|$ as $[\hat{Q}^a]_{f,g} = \hat{p}(f, g \mid a)$.

Step 1: $\forall a \in \mathcal{I}$, use the algorithm in figure 2 with input $\hat{Q}^a$ to derive estimates $\hat{r}(f \mid h, a)$ and $\hat{s}(g \mid h, a)$.

Remark: These quantities are estimates of $P(F_1 = f \mid H_1 = h, A = a)$ and $P(G = g \mid H = h, A = a)$ respectively. Note that under the independence assumptions of the L-PCFG,
$P(F_1 = f \mid H_1 = h, A = a) = P(F_2 = f \mid H_2 = h, A_2 = a) = P(F_3 = f \mid H_3 = h, A_3 = a)$.

Step 2: For each rule $a \rightarrow b\ c$, find $\hat{t}(h_1, h_2, h_3 \mid a \rightarrow b\ c)$ values that maximize

$$\sum_{g, f_2, f_3} \hat{p}(g, f_2, f_3 \mid a \rightarrow b\ c) \log \sum_{h_1, h_2, h_3} \hat{t}(h_1, h_2, h_3 \mid a \rightarrow b\ c) \hat{s}(g \mid h_1, a) \hat{r}(f_2 \mid h_2, b) \hat{r}(f_3 \mid h_3, c) \quad (1)$$

under the constraints $\hat{t}(h_1, h_2, h_3 \mid a \rightarrow b\ c) \geq 0$, and $\sum_{h_1, h_2, h_3} \hat{t}(h_1, h_2, h_3 \mid a \rightarrow b\ c) = 1$.

Remark: the function in Eq. 1 is concave in the values $\hat{t}$ being optimized over. We use the EM algorithm, which converges to a global optimum.

Step 3: $\forall a \rightarrow b\ c, h_1, h_2, h_3$, calculate rule parameters as follows:

$$\hat{t}(a \rightarrow b\ c, h_2, h_3 \mid a, h_1) = \hat{t}(a \rightarrow b\ c, h_1, h_2, h_3 \mid a) / \sum_{b, c, h_2, h_3} \hat{t}(a \rightarrow b\ c, h_1, h_2, h_3 \mid a)$$

where $\hat{t}(a \rightarrow b\ c, h_1, h_2, h_3 \mid a) = \hat{p}(a \rightarrow b\ c \mid a) \times \hat{t}(h_1, h_2, h_3 \mid a \rightarrow b\ c)$.

Output: Parameter estimates $\hat{t}(a \rightarrow b\ c, h_2, h_3 \mid a, h_1)$ for all rules $a \rightarrow b\ c$, for all $(h_1, h_2, h_3) \in [m] \times [m] \times [m]$.

Figure 1: The learning algorithm for the $t(a \rightarrow b\ c, h_1, h_2, h_3 \mid a)$ parameters of an L-PCFG.

the estimates $\hat{p}$ in assumption 3 to have converged to the correct underlying values. A more detailed analysis of the algorithm would derive sample complexity results, giving guarantees on the sample size M required to reach a level of accuracy ϵ in the estimates, with probability at least $1 - \delta$, as a function of ϵ, δ, and other relevant quantities such as $n, d, d', m, \alpha, \beta$ and so on.

In spite of the strength of the infinite data assumption, we stress the importance of this result as a guarantee for the algorithm. First, a guarantee of correct parameter values in the limit of infinite data is typically the starting point for a sample complexity result (see for example (Hsu et al., 2009; Anandkumar et al., 2012)). Second, our sense is that a sample complexity result can be derived for our algorithm using standard methods: specifically, the analysis in (Arora et al., 2012) gives one set of guarantees; the remaining optimization problems we solve are convex maximum-likelihood problems, which are also relatively easy to analyze. Note that several pieces of previous work on spectral methods for latent-variable models focus on algorithms that are correct under the infinite data assumption.

5 The Matrix Decomposition Algorithm

This section describes the matrix decomposition algorithm used in Step 1 of the learning algorithm.

5.1 Problem Setting

Our goal will be to solve the following matrix decomposition problem:

Matrix Decomposition Problem (MDP) 1. *Design an algorithm with the following inputs, assumptions, and outputs:*

Inputs: *Integers m, d and d', and a matrix $Q \in \mathbb{R}^{d \times d'}$ such that $Q_{f,g} = \sum_{h=1}^{m} \pi(h) r(f \mid h) s(g \mid h)$ for some unknown parameters $\pi(h)$, $r(f \mid h)$ and $s(g \mid h)$ satisfying:*
1) $\pi(h) \geq 0$, $\sum_{h=1}^{m} \pi(h) = 1$;
2) $r(f \mid h) \geq 0$, $\sum_{f=1}^{d} r(f \mid h) = 1$;
3) $s(g \mid h) \geq 0$, $\sum_{g=1}^{d'} s(g \mid h) = 1$.
Assumptions: *There are values $\alpha > 0$ and $\beta > 0$ such that the r parameters of the model are α-separable, and the s parameters of the model are β-separable.*
Outputs: *Estimates $\hat{\pi}(h)$, $\hat{r}(f \mid h)$ and $\hat{s}(g \mid h)$ such that there is some permutation $\sigma : [m] \rightarrow [m]$ such that $\forall h, \ \hat{\pi}(h) = \pi(\sigma(h))$, $\forall f, h, \quad \hat{r}(f \mid h) = r(f \mid \sigma(h))$, and $\forall g, h, \quad \hat{s}(g \mid h) = s(g \mid \sigma(h))$.*

The definition of α-separability is as follows (β-separability for $s(g \mid h)$ is analogous):

Definition 2 (α-separability)**.** *The parameters $r(f \mid h)$ are α-separable if for all $h \in [m]$, there is some $j \in [d]$ such that: 1) $r(j \mid h) \geq \alpha$; and 2) $r(j \mid h') = 0$ for $h' \neq h$.*

This matrix decomposition problem has clear relevance to problems in learning of latent-variable models, and in particular is a core step of the algorithm in figure 1. When given a matrix $\hat{Q}^a$ with entries $\hat{Q}^a_{f,g} = \sum_h p(h \mid a) p(f \mid h, a) p(g \mid h, a)$, where $p(\ldots)$ refers to a distribution derived from an underlying L-PCFG which satisfies the pivot assumption, the method will recover the values for $p(h \mid a)$, $p(f \mid h, a)$ and $p(g \mid h, a)$ up to a permutation over the latent states.

5.2 The Algorithm of Arora et al. (2013)

This section describes a variant of the algorithm of Arora et al. (2013), which is used as a component of our algorithm for MDP 1. One of the properties of this algorithm is that it solves the following problem:

Matrix Decomposition Problem (MDP) 2. *Design an algorithm with the following inputs, assumptions, and outputs:*
Inputs: *Same as matrix decomposition problem 1.*
Assumptions: *The parameters $r(f \mid h)$ of the model are α-separable for some value $\alpha > 0$.*
Outputs: *Estimates $\hat{\pi}(h)$ and $\hat{r}(f \mid h)$ such that $\exists \sigma : [m] \rightarrow [m]$ such that $\forall h, \ \hat{\pi}(h) = \pi(\sigma(h))$, $\forall f, h, \quad \hat{r}(f \mid h) = r(f \mid \sigma(h))$.*

This is identical to Matrix Decomposition Problem 1, but without the requirement that the values $s(g \mid h)$ are returned by the algorithm. Thus an algorithm that solves MDP 2 in some sense solves "one half" of MDP 1.

For completeness we give a sketch of the algorithm that we use; it is inspired by the algorithm of Arora et al. (2012), but has some important differences. The algorithm is as follows:

Step 1: Derive a function $\phi : [d'] \rightarrow \mathbb{R}^l$ that maps each integer $g \in [d']$ to a representation $\phi(g) \in \mathbb{R}^l$. The integer l is typically much smaller than d', implying that the representation is of low dimension. Arora et al. (2012) derive ϕ as a random projection with a carefully chosen dimension l. In our experiments, we use canonical correlation analysis (CCA) on the matrix Q to give a representation $\phi(g) \in \mathbb{R}^l$ where $l = m$.

Step 2: For each $f \in [d]$, calculate

$$v_f \ = \ \mathbf{E}[\phi(g) \mid f] = \sum_{g=1}^{d'} p(g \mid f) \phi(g)$$

where $p(g \mid f) = Q_{f,g} / \sum_g Q_{f,g}$. It follows that

$$v_f = \sum_{g=1}^{d'} \sum_{h=1}^{m} p(h \mid f) p(g \mid h) \phi(g) = \sum_{h=1}^{m} p(h \mid f) w_h$$

where $w_h \in \mathbb{R}^l$ is equal to $\sum_{g=1}^{d'} p(g \mid h) \phi(g)$.

Hence the v_f vectors lie in the convex hull of a set of vectors $w_1 \ldots w_m \in \mathbb{R}^l$. Crucially, for any pivot word f for latent state h, we have $p(h \mid f) = 1$, hence $v_f = w_h$. *Thus by the pivot assumption, the set of points $v_1 \ldots v_d$ includes the vertices of the convex hull. Each point v_j is a convex combination of the vertices $w_1 \ldots w_m$, where the weights in this combination are equal to $p(h \mid j)$.*

Step 3: Use the `FastAnchorWords` algorithm of (Arora et al., 2012) to identify m vectors $v_{s_1}, v_{s_2}, \ldots v_{s_m}$. The `FastAnchorWords` algorithm has the guarantee that there is a permutation $\sigma : [m] \rightarrow [m]$ such that $v_{s_i} = w_{\sigma(i)}$ for all i. This algorithm recovers the vertices of the convex hull described in step 2, using a method that greedily picks points that are as far as possible from the subspace spanned by previously picked points.

Step 4: For each $f \in [d]$ solve the problem

$$\arg \min_{\gamma_1, \gamma_2, \ldots, \gamma_m} || \sum_h \gamma_h v_{s_h} - v_f ||^2$$

subject to $\gamma_h \geq 0$ and $\sum_h \gamma_h = 1$. We use the algorithm of (Frank and Wolfe, 1956; Clarkson, 2010) for this purpose. Set $q(h \mid f) = \gamma_h$.

Return the final quantities:

$$\hat{\pi}(h) = \sum_f p(f)q(h|f) \quad \hat{r}(f|h) = \frac{p(f)q(h|f)}{\sum_f p(f)q(h|f)}$$

where $p(f) = \sum_g Q_{f,g}$.

5.3 An Algorithm for MDP 1

Figure 2 shows an algorithm that solves MDP 1. In steps 1 and 2 of the algorithm, the algorithm of section 5.2 is used to recover estimates $\hat{r}(f \mid h)$ and $\hat{s}(g \mid h)$. These distributions are equal to $p(f \mid h)$ and $p(g \mid h)$ up to permutations σ and σ' of the latent states respectively; unfortunately there is no guarantee that σ and σ' are the same permutation. Step 3 estimates parameters $t(h' \mid h)$ that effectively map the permutation implied by $\hat{r}(f \mid h)$ to the permutation implied by $\hat{s}(g \mid h)$; the latter distribution is recalculated as $\sum_{h'} \hat{t}(h' \mid h)\hat{s}(g \mid h')$.

We now state the following theorem:

Theorem 2. *The algorithm in figure 2 solves Matrix Decomposition Problem 1.*

Proof: See the supplementary material.

Remark: A natural alternative to the algorithm presented would be to run Step 1 of the original algorithm, but to replace steps 2 and 3 with a step that finds $\hat{s}(g \mid h)$ values that maximize

$$\sum_{f,g} Q_{f,g} \log \sum_h \hat{r}(h \mid f)\hat{s}(g \mid h)$$

This is again a convex optimization problem. We may explore this algorithm in future work.

6 Additional Details of the Algorithm

6.1 Recovery of the π and q Parameters

The recovery of the π and q parameters relies on the following additional (but benign) assumptions on the functions τ and ρ:

1) For any inside tree t such that t is a unary rule of the form $a \rightarrow x$, the function τ is defined as $\tau(t) = t$.[3]

2) The set of outside tree features $\mathcal{G}$ contains a special symbol $\square$, and $g(o) = \square$ if and only if the outside tree o is derived from a non-terminal node at the root of a skeletal tree.

[3]Note that if other features on unary rules are desired, we can use multiple feature functions $\tau^1(t) \ldots \tau^K(t)$, where $\tau^1(t) = t$ for inside trees, and the functions $\tau^2(t) \ldots \tau^K(t)$ define other features.

Inputs: As in Matrix Decomposition Problem 1.

Assumptions: As in Matrix Decomposition Problem 1.

Algorithm:

Step 1. Run the algorithm of section 5.2 on the matrix Q to derive estimates $\hat{r}(f \mid h)$ and $\hat{\pi}(h)$. Note that under the guarantees of the algorithm, there is some permutation σ such that $\hat{r}(f \mid h) = r(f \mid \sigma(h))$. Define

$$\hat{r}(h \mid f) = \frac{\hat{r}(f \mid h)\hat{\pi}(h)}{\sum_h \hat{r}(f \mid h)\hat{\pi}(h)}$$

Step 2. Run the algorithm of section 5.2 on the matrix $Q^\top$ to derive estimates $\hat{s}(g \mid h)$. Under the guarantees of the algorithm, there is some permutation σ' such that $\hat{s}(g \mid h) = s(g \mid \sigma'(h))$. Note however that it is not necessarily the case that $\sigma = \sigma'$.

Step 3. Find $\hat{t}(h' \mid h)$ for all $h, h' \in [m]$ that maximize

$$\sum_{f,g} Q_{f,g} \log \sum_{h,h'} \hat{r}(h \mid f)\hat{t}(h' \mid h)\hat{s}(g \mid h') \qquad (3)$$

subject to $\hat{t}(h' \mid h) \geq 0$, and $\forall h,\ \sum_{h'} \hat{t}(h' \mid h) = 1$.

Remark: the function in Eq. 3 is concave in the $\hat{t}$ parameters. We use the EM algorithm to find a global optimum.

Step 4. Return the following values:

- $\hat{\pi}(h)$ for all h, as an estimate of $\pi(\sigma(h))$ for some permutation σ.
- $\hat{r}(f \mid h)$ for all f, h as an estimate of $r(f \mid \sigma(h))$ for the same permutation σ.
- $\sum_{h'} \hat{t}(h' \mid h)\hat{s}(g \mid h')$ as an estimate of $s(f \mid \sigma(h))$ for the same permutation σ.

Figure 2: The algorithm for Matrix Decomposition Problem 1

Under these assumptions, the algorithm in figure 1 recovers estimates $\hat{\pi}(a, h)$ and $\hat{q}(a \rightarrow x \mid a, h)$. Simply set

$$\hat{q}(a \rightarrow x \mid a, h) = \hat{r}(f \mid h, a) \text{ where } f = a \rightarrow x$$

and $\hat{\pi}(a, h) = \hat{p}(\square, h, a)/\sum_{h,a} \hat{p}(\square, h, a)$ where $\hat{p}(\square, h, a) = \hat{g}(\square \mid h, a)\hat{p}(h \mid a)\hat{p}(a)$. Note that $\hat{p}(h \mid a)$ can be derived from the matrix decomposition step when applied to $\hat{Q}^a$, and $\hat{p}(a)$ is easily recovered from the training examples.

6.2 Extension to Include Multiple Features

We now describe an extension to allow K separate functions $\tau^{(k)}(t)$ for $k = 1 \ldots K$ mapping inside trees to features, and L feature functions $\rho^{(l)}(o)$ for $l = 1 \ldots L$ over outside trees.

The algorithm in figure 1 can be extended as follows. First, Step 1 of the algorithm (the matrix

decomposition step) can be extended to provide estimates $\hat{r}^{(k)}(f^{(k)} \mid h, a)$ and $\hat{s}^{(l)}(g^{(l)} \mid h, a)$. In brief, this involves running CCA on a matrix $\mathbf{E}[\phi(T)(\psi(O))^\top \mid A = a]$ where ϕ and ψ are inside and outside binary feature vectors derived directly from the inside and outside features, using a one-hot representation. CCA results in a low-dimensional representation that can be used in the steps described in section 5.2; the remainder of the algorithm is the same. In practice, the addition of multiple features may lead to better CCA representations.

Next, we modify the objective function in Eq. 1 to be the following:

$$\sum_{i,j,k} \sum_{g^i, f_2^j, f_3^k} p(g^i, f_2^j, f_3^k \mid a \rightarrow b\ c) \log \kappa^{i,j,k}(g^i, f_2^j, f_3^k)$$

where

$$\begin{aligned} &\kappa^{i,j,k}(g^i, f_2^j, f_3^k) \\ &= \sum_{h_1,h_2,h_3} \Big(\hat{t}(h_1, h_2, h_3 \mid a \rightarrow b\ c) \\ &\quad \times \hat{s}^i(g^i \mid h_1, a)\hat{r}^j(f_2^j \mid h_2, b)\hat{r}^k(f_3^k \mid h_3, c)\Big) \end{aligned}$$

Thus the new objective function consists of a sum of $L \times M^2$ terms, each corresponding to a different combination of inside and outside features. The function remains concave.

6.3 Use as an Initializer for EM

The learning algorithm for L-PCFGs can be used as an initializer for the EM algorithm for L-PCFGs. Two-step estimation methods such as these are well known in statistics; there are guarantees for example that if the first estimator is consistent, and the second step finds the closest local maxima of the likelihood function, then the resulting estimator is both consistent and efficient (in terms of number of samples required). See for example page 453 or Theorem 4.3 (page 454) of (Lehmann and Casella, 1998).

7 Experiments on Parsing

This section describes parsing experiments using the learning algorithm for L-PCFGs. We use the Penn WSJ treebank (Marcus et al., 1993) for our experiments. Sections 2–21 were used as training data, and sections 0 and 22 were used as development data. Section 23 was used as the test set.

The experimental setup is the same as described by Cohen et al. (2013). The trees are binarized (Petrov et al., 2006) and for the EM algorithm we use the initialization method described in Matsuzaki et al. (2005). For the pivot algorithm we use multiple features $\tau^1(t) \ldots \tau^K(t)$ and $\rho^1(o) \ldots \rho^L(o)$ over inside and outside trees, using the features described by Cohen et al. (2013).

	sec. 22				sec. 23
m	8	16	24	32	
EM	86.69 40	88.32 30	88.35 30	88.56 20	87.76
Spectral	85.60	87.77	88.53	88.82	88.05
Pivot	83.56	86.00	86.87	86.40	85.83
Pivot+EM	86.83 2	88.14 6	88.64 2	88.55 2	88.03

Table 1: Results on the development data (section 22) and test data (section 23) for various learning algorithms for L-PCFGs. For EM and pivot+EM experiments, the second line denotes the number of iterations required to reach the given optimal performance on development data. Results for section 23 are used with the best model for section 22 in the corresponding row. The results for EM and spectral are reported from Cohen et al. (2013).

Table 1 gives the F1 accuracy on the development and test sets for the following methods:

EM: The EM algorithm as used by Matsuzaki et al. (2005) and Petrov et al. (2006).

Spectral: The spectral algorithm of Cohen et al. (2012) and Cohen et al. (2013).

Pivot: The algorithm described in this paper.

Pivot+EM: The algorithm described in this paper, followed by 1 or more iterations of the EM algorithm with parameters initialized by the pivot algorithm. (See section 6.3.)

For the EM and Pivot+EM algorithms, we give the number of iterations of EM required to reach optimal performance on the development data.

The results show that the EM, Spectral, and Pivot+EM algorithms all perform at a very similar level of accuracy. The Pivot+EM results show that very few EM iterations—just 2 iterations in most conditions—are required to reach optimal performance when the Pivot model is used as an initializer for EM. The Pivot results lag behind the Pivot+EM results by around 2-3%, but they are close enough to optimality to require very few EM iterations when used as an initializer.

8 Experiments on the Saul and Pereira (1997) Model for Language Modeling

We now describe a second set of experiments, on the Saul and Pereira (1997) model for language modeling. Define V to be the set of words in the vocabulary. For any $w_1, w_2 \in V$, the Saul and Pereira (1997) model then defines $p(w_2 \mid w_1) = \sum_{h=1}^{m} r(h \mid w_1)s(w_2 \mid h)$ where $r(h \mid w_1)$ and

	Brown								NYT							
m	2	4	8	16	32	128	256	test	2	4	8	16	32	128	256	test
EM	**737** 14	599 14	**488** 19	468 12	430 10	388 9	365 8	364	926 36	**733** 39	562 42	**420** 33	**361** 38	284 35	**265** 32	267
bi-KN +int.	408							415	271							279
tri-KN+int.	386							394	150							158
pivot	852	718	605	559	537	426	597	560	1227	1264	896	717	738	782	886	715
pivot+EM	758 2	**582** 3	502 2	**425** 1	**374** 1	**310** 1	**327** 1	357	**898** 20	754 14	**553** 13	441 15	394 10	**279** 19	292 12	281

Table 2: Language model perplexity with the Brown corpus and the Gigaword corpus (New York Times portion) for the second half of the development set, and the test set. With EM and Pivot+EM, the number of iterations for EM to reach convergence is given below the perplexity. The best result for each column (for each m value) is in bold. The "test" column gives perplexity results on the test set. Each perplexity calculation on the test set is done using the best model on the development set. bi-KN+int and tri-KN+int are bigram and trigram Kneser-Ney interpolated models (Kneser and Ney, 1995), using the SRILM toolkit.

$s(w_2 \mid h)$ are parameters of the approach. The conventional approach to estimation of the parameters $r(h \mid w_1)$ and $s(w_2 \mid h)$ from a corpus is to use the EM algorithm. In this section we compare the EM algorithm to a pivot-based method. It is straightforward to represent this model as an L-PCFG, and hence to use our implementation for estimation.

In this special case, the L-PCFG learning algorithm is equivalent to a simple algorithm, with the following steps: 1) define the matrix Q with entries $Q_{w_1,w_2} = \text{count}(w_1, w_2)/N$ where $\text{count}(w_1, w_2)$ is the number of times that bigram (w_1, w_2) is seen in the data, and $N = \sum_{w_1,w_2} \text{count}(w_1, w_2)$. Run the algorithm of section 5.2 on Q to recover estimates $\hat{s}(w_2 \mid h)$; 2) estimate $\hat{r}(h \mid w_1)$ using the EM algorithm to optimize the function $\sum_{w_1,w_2} Q_{w_1,w_2} \log \sum_h \hat{r}(h \mid w_1)\hat{s}(w_2 \mid h)$ with respect to the $\hat{r}$ parameters; this function is concave in these parameters.

We performed the language modeling experiments for a number of reasons. First, because in this case the L-PCFG algorithm reduces to a simple algorithm, it allows us to evaluate the core ideas in the method very directly. Second, it allows us to test the pivot method on the very large datasets that are available for language modeling.

We use two corpora for our experiments. The first is the Brown corpus, as used by Bengio et al. (2006) in language modeling experiments. Following Bengio et al. (2006), we use the first 800K words for training (and replace all words that appear once with an UNK token), the next 200K words for development, and the remaining data (165,171 tokens) as a test set. The size of the vocabulary is 24,488 words. The second corpus we use is the New York Times portion of the Gigaword corpus. Here, the training set consists of 1.31 billion tokens. We use 159 million tokens for development set and 156 million tokens for test. All words that appeared less than 20 times in the training set were replaced with the UNK token. The size of the vocabulary is 235,223 words. Unknown words in test data are ignored when calculating perplexity (this is the standard set-up in the SRILM toolkit).

In our experiments we use the first half of each development set to optimize the number of iterations of the EM or Pivot+EM algorithms. As before, Pivot+EM uses 1 or more EM steps with parameter initialization from the Pivot method.

Table 2 gives perplexity results for the different algorithms. As in the parsing experiments, the Pivot method alone performs worse than EM, but the Pivot+EM method gives results that are competitive with EM. The Pivot+EM method requires fewer iterations of EM than the EM algorithm. On the Brown corpus the difference is quite dramatic, with only 1 or 2 iterations required, as opposed to 10 or more for EM. For the NYT corpus the Pivot+EM method requires more iterations (around 10 or 20), but still requires significantly fewer iterations than the EM algorithm.

On the Gigaword corpus, with $m = 256$, EM takes 12h57m (32 iterations at 24m18s per iteration) compared to 1h50m for the Pivot method. On Brown, EM takes 1m47s (8 iterations) compared to 5m44s for the Pivot method. Both the EM and pivot algorithm implementations were highly optimized, and written in Matlab. Results at other values of m are similar. From these results the Pivot method appears to become more competitive speed-wise as the data size increases (the Gigaword corpus is more than 1,300 times larger than the Brown corpus).

9 Conclusion

We have described a new algorithm for parameter estimation in L-PCFGs. The algorithm is provably correct, and performs well in practice when used in conjunction with EM.

References

A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky. 2012. Tensor decompositions for learning latent-variable models. arXiv:1210.7559.

S. Arora, R. Ge, and A. Moitra. 2012. Learning topic models–going beyond SVD. In *Proceedings of FOCS*.

S. Arora, R. Ge, Y. Halpern, D. M. Mimno, A. Moitra, D. Sontag, Y. Wu, and M. Zhu. 2013. A practical algorithm for topic modeling with provable guarantees. In *Proceedings of ICML*.

R. Bailly, A. Habrar, and F. Denis. 2010. A spectral approach for probabilistic grammatical inference on trees. In *Proceedings of ALT*.

B. Balle, A. Quattoni, and X. Carreras. 2011. A spectral learning algorithm for finite state transducers. In *Proceedings of ECML*.

Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, and J.-L. Gauvain. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer.

K. L. Clarkson. 2010. Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. *ACM Transactions on Algorithms (TALG)*, 6(4):63.

S. B. Cohen, K. Stratos, M. Collins, D. F. Foster, and L. Ungar. 2012. Spectral learning of latent-variable PCFGs. In *Proceedings of ACL*.

S. B. Cohen, K. Stratos, M. Collins, D. P. Foster, and L. Ungar. 2013. Experiments with spectral learning of latent-variable PCFGs. In *Proceedings of NAACL*.

A. Dempster, N. Laird, and D. Rubin. 1977. Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38.

P. Dhillon, J. Rodu, M. Collins, D. P. Foster, and L. H. Ungar. 2012. Spectral dependency parsing with latent variables. In *Proceedings of EMNLP*.

M. Frank and P. Wolfe. 1956. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110.

H. Hotelling. 1936. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377.

D. Hsu, S. M. Kakade, and T. Zhang. 2009. A spectral algorithm for learning hidden Markov models. In *Proceedings of COLT*.

R. Kneser and H. Ney. 1995. Improved backing-off for m-gram language modeling. In *Proceedings of ICASSP*.

E. L. Lehmann and G. Casella. 1998. *Theory of Point Estimation (Second edition)*. Springer.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19:313–330.

T. Matsuzaki, Y. Miyao, and J. Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of ACL*.

A. Parikh, L. Song, and E. P. Xing. 2011. A spectral algorithm for latent tree graphical models. In *Proceedings of ICML*.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of COLING-ACL*.

L. Saul and F. Pereira. 1997. Aggregate and mixed-order markov models for statistical language processing. In *Proceedings of EMNLP*.

S. Siddiqi, B. Boots, and G. Gordon. 2010. Reduced-rank hidden markov models. *Journal of Machine Learning Research*, 9:741–748.

Spectral Unsupervised Parsing with Additive Tree Metrics

Ankur P. Parikh
School of Computer Science
Carnegie Mellon University
apparikh@cs.cmu.edu

Shay B. Cohen
School of Informatics
University of Edinburgh
scohen@inf.ed.ac.uk

Eric P. Xing
School of Computer Science
Carnegie Mellon University
epxing@cs.cmu.edu

Abstract

We propose a spectral approach for unsupervised constituent parsing that comes with theoretical guarantees on latent structure recovery. Our approach is grammarless – we directly learn the bracketing structure of a given sentence without using a grammar model. The main algorithm is based on lifting the concept of additive tree metrics for structure learning of latent trees in the phylogenetic and machine learning communities to the case where the tree structure varies across examples. Although finding the "minimal" latent tree is NP-hard in general, for the case of projective trees we find that it can be found using bilexical parsing algorithms. Empirically, our algorithm performs favorably compared to the constituent context model of Klein and Manning (2002) without the need for careful initialization.

1 Introduction

Solutions to the problem of grammar induction have been long sought after since the early days of computational linguistics and are interesting both from cognitive and engineering perspectives. Cognitively, it is more plausible to assume that children obtain only terminal strings of parse trees and not the actual parse trees. This means the unsupervised setting is a better model for studying language acquisition. From the engineering perspective, training data for unsupervised parsing exists in abundance (i.e. sentences and part-of-speech tags), and is much cheaper than the syntactically annotated data required for supervised training.

Most existing solutions treat the problem of unsupervised parsing by assuming a generative process over parse trees e.g. probabilistic context free grammars (Jelinek et al., 1992), and the constituent context model (Klein and Manning, 2002). Learning then reduces to finding a set of parameters that are estimated by identifying a local maximum of an objective function such as the likelihood (Klein and Manning, 2002) or a variant of it (Smith and Eisner, 2005; Cohen and Smith, 2009; Headden et al., 2009; Spitkovsky et al., 2010b; Gillenwater et al., 2010; Golland et al., 2012). Unfortunately, finding the global maximum for these objective functions is usually intractable (Cohen and Smith, 2012) which often leads to severe local optima problems (but see Gormley and Eisner, 2013). Thus, strong experimental results are often achieved by initialization techniques (Klein and Manning, 2002; Gimpel and Smith, 2012), incremental dataset use (Spitkovsky et al., 2010a) and other specialized techniques to avoid local optima such as count transforms (Spitkovsky et al., 2013). These approaches, while empirically promising, generally lack theoretical justification.

On the other hand, recently proposed spectral methods approach the problem via restriction of the PCFG model (Hsu et al., 2012) or matrix completion (Bailly et al., 2013). These novel perspectives offer strong theoretical guarantees but are not designed to achieve competitive empirical results.

In this paper, we suggest a different approach, to provide a first step to bridging this theory-experiment gap. More specifically, we approach unsupervised constituent parsing from the perspective of *structure learning* as opposed to parameter learning. We associate each sentence with an undirected latent tree graphical model, which is a tree consisting of both observed variables (corresponding to the words in the sentence) and an additional set of latent variables that are unobserved in the data. This undirected latent tree is then directed via a *direction mapping* to give the final constituent parse.

In our framework, parsing reduces to finding the best latent structure for a given sentence. However, due to the presence of latent variables, structure learning of latent trees is substantially more complicated than in observed models. As before, one solution would be local search heuristics.

Intuitively, however, latent tree models encode low rank dependencies among the observed variables permitting the development of "spec-

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1062–1072,
Baltimore, Maryland, USA, June 23-25 2014.

tral" methods that can lead to provably correct solutions. In particular we leverage the concept of additive tree metrics (Buneman, 1971; Buneman, 1974) in phylogenetics and machine learning that can create a special distance metric among the observed variables as a function of the underlying spectral dependencies (Choi et al., 2011; Song et al., 2011; Anandkumar et al., 2011; Ishteva et al., 2012). Additive tree metrics can be leveraged by "meta-algorithms" such as neighbor-joining (Saitou and Nei, 1987) and recursive grouping (Choi et al., 2011) to provide consistent learning algorithms for latent trees.

Moreover, we show that it is desirable to learn the "minimal" latent tree based on the tree metric ("minimum evolution" in phylogenetics). While this criterion is in general NP-hard (Desper and Gascuel, 2005), for projective trees we find that a bilexical parsing algorithm can be used to find an exact solution efficiently (Eisner and Satta, 1999).

Unlike in phylogenetics and graphical models, where a single latent tree is constructed for all the data, in our case, each part of speech sequence is associated with its own parse tree. This leads to a severe data sparsity problem even for moderately long sentences. To handle this issue, we present a strategy that is inspired by ideas from kernel smoothing in the statistics community (Zhou et al., 2010; Kolar et al., 2010b; Kolar et al., 2010a). This allows principled sharing of samples from different but similar underlying distributions.

We provide theoretical guarantees on the recovery of the correct underlying latent tree and characterize the associated sample complexity under our technique. Empirically we evaluate our method on data in English, German and Chinese. Our algorithm performs favorably to Klein and Manning's (2002) constituent-context model (CCM), without the need for careful initialization. In addition, we also analyze CCM's sensitivity to initialization, and compare our results to Seginer's algorithm (Seginer, 2007).

2 Learning Setting and Model

In this section, we detail the learning setting and a conditional tree model we learn the structure for.

2.1 Learning Setting

Let $\boldsymbol{w} = (w_1, ..., w_\ell)$ be a vector of words corresponding to a sentence of length ℓ. Each w_i is represented by a vector in $\mathbb{R}^p$ for $p \in \mathbb{N}$. The vector is an embedding of the word in some space, chosen from a fixed dictionary that maps word types to $\mathbb{R}^p$. In addition, let $\boldsymbol{x} = (x_1, ..., x_\ell)$ be the associated vector of part-of-speech (POS) tags (i.e. x_i is the POS tag of w_i).

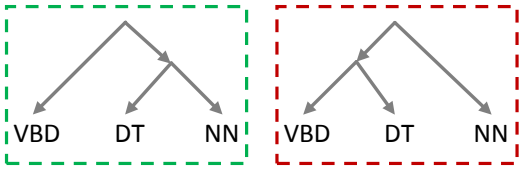


Figure 2: Candidate constituent parses for $\boldsymbol{x} = (\texttt{VBD}, \texttt{DT}, \texttt{NN})$ (left-correct, right -incorrect)

In our learning algorithm, we assume that examples of the form $(\boldsymbol{w}^{(i)}, \boldsymbol{x}^{(i)})$ for $i \in [N] = \{1, \ldots, N\}$ are given, and the goal is to predict a bracketing parse tree for each of these examples. The word embeddings are used during the learning process, but the final decoder that the learning algorithm outputs maps a POS tag sequence $\boldsymbol{x}$ to a parse tree. While ideally we would want to use the word information in decoding as well, much of the syntax of a sentence is determined by the POS tags, and relatively high level of accuracy can be achieved by learning, for example, a supervised parser from POS tag sequences.

Just like our decoder, our model assumes that the bracketing of a given sentence is a function of its POS tags. The POS tags are generated from some distribution, followed by a deterministic generation of the bracketing parse tree. Then, latent states are generated for each bracket, and finally, the latent states at the yield of the bracketing parse tree generate the words of the sentence (in the form of embeddings). The latent states are represented by vectors $z \in \mathbb{R}^m$ where $m < p$.

2.2 Intuition

For intuition, consider the simple tag sequence $\boldsymbol{x} = (\texttt{VBD}, \texttt{DT}, \texttt{NN})$. Two candidate constituent parse structures are shown in Figure 2 and the correct one is boxed in green (the other in red). Recall that our training data contains word phrases that have the tag sequence $\boldsymbol{x}$ e.g. $\boldsymbol{w}^{(1)} = (\texttt{hit}, \texttt{the}, \texttt{ball})$, $\boldsymbol{w}^{(2)} = (\texttt{ate}, \texttt{an}, \texttt{apple})$.

Intuitively, the words in the above phrases exhibit dependencies that can reveal the parse structure. The determiner (w_2) and the direct object (w_3) are correlated in that the choice of determiner depends on the plurality of w_3. However, the choice of verb (w_1) is mostly independent of the determiner. We could thus conclude that w_2 and w_3 should be closer in the parse tree than w_1

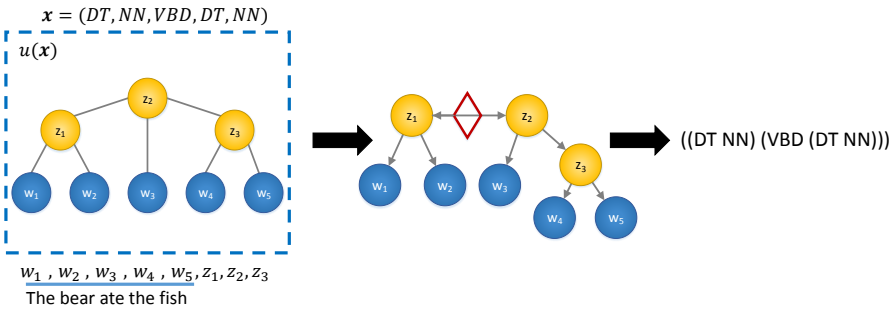


Figure 1: Example for the tag sequence (DT, NN, VBD, DT, NN) showing the overview of our approach. We first learn a undirected latent tree for the sequence (left). We then apply a direction mapping h_{dir} to direct the latent tree (center). This can then easily be converted into a bracketing (right).

and w_2, giving us the correct structure. Informally, the latent state z corresponding to the (w_2, w_3) bracket would store information about the plurality of z, the key to the dependence between w_2 and w_3. It would then be reasonable to assume that w_2 and w_3 are independent given z.

2.3 A Conditional Latent Tree Model

Following this intuition, we propose to model the distribution over the latent bracketing states and words for each tag sequence $\boldsymbol{x}$ as a latent tree graphical model, which encodes conditional independences among the words given the latent states.

Let $\mathcal{V} := \{w_1, ..., w_\ell, z_1, ..., z_H\}$, with w_i representing the word embeddings, and z_i representing the latent states of the bracketings. Then, according to our base model it holds that:

$$p(\boldsymbol{w}, \boldsymbol{z}|\boldsymbol{x}) = \prod_{i=1}^{H} p(z_i|\pi_{\boldsymbol{x}}(z_i), \theta(\boldsymbol{x})) \times \prod_{i=1}^{\ell(\boldsymbol{x})} p(w_i|\pi_{\boldsymbol{x}}(w_i), \theta(\boldsymbol{x})) \quad (1)$$

where $\pi_{\boldsymbol{x}}(\cdot)$ returns the parent node index of the argument in the latent tree corresponding to tag sequence $\boldsymbol{x}$.[1] If z is the root, then $\pi_{\boldsymbol{x}}(z) = \emptyset$. All the w_i are assumed to be leaves while all the z_i are internal (i.e. non-leaf) nodes. The parameters $\theta(\boldsymbol{x})$ control the conditional probability tables. We do not commit to a certain parametric family, but see more about the assumptions we make about θ in §3.2. The parameter space is denoted Θ. The model assumes a factorization according to a latent-variable tree. The latent variables can incorporate various linguistic properties, such as head information, valence of dependency being generated, and so on. This information is expected to be learned automatically from data.

Our generative model deterministically maps a POS sequence to a bracketing via an undirected latent-variable tree. The orientation of the tree is determined by a *direction mapping* $h_{\text{dir}}(u)$, which is fixed during learning and decoding. This means our decoder first identifies (given a POS sequence) an undirected tree, and then orients it by applying h_{dir} on the resulting tree (see below).

Define $\mathcal{U}$ to be the set of undirected latent trees where all internal nodes have degree exactly 3 (i.e. they correspond to binary bracketing), and in addition $h_{\text{dir}}(u)$ for any $u \in \mathcal{U}$ is projective (explained in the h_{dir} section). In addition, let $\mathcal{T}$ be the set of binary bracketings. The complete generative model that we follow is then:

- Generate a tag sequence $\boldsymbol{x} = (x_1, \ldots, x_\ell)$
- Decide on $u(\boldsymbol{x}) \in \mathcal{U}$, the undirected latent tree that $\boldsymbol{x}$ maps to.
- Set $t \in \mathcal{T}$ by computing $t = h_{\text{dir}}(u)$.
- Set $\theta \in \Theta$ by computing $\theta = \theta(\boldsymbol{x})$.
- Generate a tuple $\boldsymbol{v} = (w_1, \ldots, w_\ell, z_1, ..., z_H)$ where $w_i \in \mathbb{R}^p, z_j \in \mathbb{R}^m$ according to Eq. 1.

See Figure 1 (left) for an example.

The Direction Mapping h_{dir}. Generating a bracketing via an undirected tree enables us to build on existing methods for structure learning of latent-tree graphical models (Choi et al., 2011; Anandkumar et al., 2011). Our learning algorithm focuses on recovering the undirected tree based for the generative model that was described above. This undirected tree is converted into a directed tree by applying h_{dir}. The mapping h_{dir} works in three steps:

- It first chooses a top bracket $([1, R-1], [R, \ell])$ where R is the mid-point of the bracket and ℓ is the length of the sentence.
- It marks the edge $e_{i,j}$ that splits the tree according to the top bracket as the "root edge" (marked in red in Figure 1(center))
- It then creates t from u by directing the tree outward from $e_{i,j}$ as shown in Figure 1(center)

[1] At this point, π refers to an arbitrary direction of the undirected tree $u(\boldsymbol{x})$.

The resulting t is a binary bracketing parse tree. As implied by the above definition of h_{dir}, selecting which edge is the root can be interpreted as determining the top bracket of the constituent parse. For example, in Figure 1, the top bracket is $([1,2],[3,5]) = ([\texttt{DT},\texttt{NN}],[\texttt{VBD},\texttt{DT},\texttt{NN}])$. Note that the "root" edge e_{z_1,z_2} partitions the leaves into precisely this bracketing. As indicated in the above section, we restrict the set of undirected trees to be those such that after applying h_{dir} the resulting t is projective i.e. there are no crossing brackets. In §4.1, we discuss an effective heuristic to find the top bracket without supervision.

3 Spectral Learning Algorithm based on Additive Tree Metrics

Our goal is to recover $t \in \mathcal{T}$ for tag sequence $\boldsymbol{x}$ using the data $\mathcal{D} = [(\boldsymbol{w}^{(i)}, \boldsymbol{x}^{(i)})]_{i=1}^{N}$. To get an intuition about the algorithm, consider a partition of the set of examples $\mathcal{D}$ into $\mathcal{D}(\boldsymbol{x}) = \{(\boldsymbol{w}^{(i)}, \boldsymbol{x}^{(i)}) \in \mathcal{D} | \boldsymbol{x}^{(i)} = \boldsymbol{x}\}$, i.e. each section in the partition has an identical sequence of part of speech tags. Assume for this section $|\mathcal{D}(\boldsymbol{x})|$ is large (we address the data sparsity issue in §3.4).

We can then proceed by learning how to map a POS sequence $\boldsymbol{x}$ to a tree $t \in \mathcal{T}$ (through $u \in \mathcal{U}$) by focusing only on examples in $\mathcal{D}(\boldsymbol{x})$.

Directly attempting to maximize the likelihood unfortunately results in an intractable optimization problem and greedy heuristics are often employed (Harmeling and Williams, 2011). Instead we propose a method that is provably consistent and returns a tree that can be mapped to a bracketing using h_{dir}.

If all the variables were observed, then the Chow-Liu algorithm (Chow and Liu, 1968) could be used to find the most likely tree structure $u \in \mathcal{U}$. The Chow-Liu algorithm essentially computes the distances among all pairs of variables (the negative of the mutual information) and then finds the minimum cost tree. However, the fact that the z_i are latent variables makes this strategy substantially more complicated. In particular, it becomes challenging to compute the distances among pairs of latent variables. What is needed is a "special" distance function that allows us to reverse engineer the distances among the latent variables given the distances among the observed variables. This is the key idea behind additive tree metrics that are the basis of our approach.

In the following sections, we describe the key steps to our method. §3.1 and §3.2 largely describe existing background on additive tree metrics and latent tree structure learning, while §3.3 and §3.4 discuss novel aspects that are unique to our problem.

3.1 Additive Tree Metrics

Let $u(\boldsymbol{x})$ be the true undirected tree of sentence $\boldsymbol{x}$ and assume the nodes $\mathcal{V}$ to be indexed by $[M] = \{1, \ldots, M\}$ such that $M = |\mathcal{V}| = H + \ell$. Furthermore, let $v \in \mathcal{V}$ refer to a node in the undirected tree (either observed or latent). We assume the existence of a distance function that allows us to compute distances between pairs of nodes. For example, as we see in §3.2 we will define the distance $d(i,j)$ to be a function of the covariance matrix $\mathbb{E}[v_i v_j^\top | u(\boldsymbol{x}), \theta(\boldsymbol{x})]$. Thus if v_i and v_j are both observed variables, the distance can be directly computed from the data.

Moreover, the metrics we construct are such that they are *tree additive*, defined below:

Definition 1 *A function $d_{u(\boldsymbol{x})} : [M] \times [M] \to \mathbb{R}$ is an* additive tree metric *(Erdős et al., 1999) for the undirected tree $u(\boldsymbol{x})$ if it is a distance metric,[2] and furthermore, $\forall i, j \in [M]$ the following relation holds:*

$$d_{u(\boldsymbol{x})}(i,j) = \sum_{(a,b) \in \text{path}_{u(\boldsymbol{x})}(i,j)} d_{u(\boldsymbol{x})}(a,b) \quad (2)$$

where $\text{path}_{u(\boldsymbol{x})}(i,j)$ is the set of all the edges in the (undirected) path from i to j in the tree $u(\boldsymbol{x})$.

As we describe below, given the tree structure, the additive tree metric property allows us to compute "backwards" the distances among the latent variables as a function of the distances among the observed variables.

Define $\boldsymbol{D}$ to be the $M \times M$ distance matrix among the M variables, i.e. $D_{ij} = d_{u(\boldsymbol{x})}(i,j)$. Let $\boldsymbol{D}_{WW}$, $\boldsymbol{D}_{ZW}$ (equal to $\boldsymbol{D}_{WZ}^\top$), and $\boldsymbol{D}_{ZZ}$ indicate the word-word, latent-word and latent-latent sub-blocks of $\boldsymbol{D}$ respectively. In addition, since $u(\boldsymbol{x})$ is assumed to be known from context, we denote $d_{u(\boldsymbol{x})}(i,j)$ just by $d(i,j)$.

Given the fact that the distance between a pair of nodes is a function of the random variables they represent (according to the true model), only $\boldsymbol{D}_{WW}$ can be empirically estimated from data. However, if the underlying tree structure is known, then Definition 1 can be leveraged to compute $\boldsymbol{D}_{ZZ}$ and $\boldsymbol{D}_{ZW}$ as we show below.

[2] This means that it satisfies $d(i,j) = 0$ if and only if $i = j$, the triangle inequality and is also symmetric.

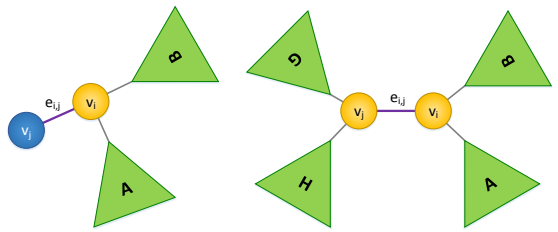


Figure 3: Two types of edges in general undirected latent trees. (a) leaf edge, (b) internal edge

We first show how to compute $d(i,j)$ for all i,j such that i and j are adjacent to each other in $u(\boldsymbol{x})$, based only on observed nodes. It then follows that the other elements of the distance matrix can be computed based on Definition 1. To show how to compute distances between adjacent nodes, consider the two cases: **(1)** (i,j) is a leaf edge; **(2)** (i,j) is an internal edge.

Case 1 (leaf edge, figure 3(a)) Assume without loss of generality that j is the leaf and i is an internal latent node. Then i must have exactly two other neighbors $a \in [M]$ and $b \in [M]$. Let A denote the set of nodes that are closer to a than i and similarly let B denote the set of nodes that are closer to b than i. Let A^* and B^* denote all the leaves (word nodes) in A and B respectively. Then using path additivity (Definition 1), it can be shown that for any $a^* \in A^*, b^* \in B^*$ it holds that:

$$d(i,j) = \frac{1}{2}\left(d(j,a^*) + d(j,b^*) - d(a^*,b^*)\right) \quad (3)$$

Note that the right-hand side only depends on distances between observed random variables.

Case 2 (internal edge, figure 3(b)) Both i and j are internal nodes. In this case, i has exactly two other neighbors $a \in [M]$ and $b \in [M]$, and similarly, j has exactly other two neighbors $g \in [M]$ and $h \in [M]$. Let A denote the set of nodes closer to a than i, and analogously for B, G, and H. Let A^*, B^*, G^*, and H^* refer to the leaves in A, B, G, and H respectively. Then for any $a^* \in A^*$, $b^* \in B^*$, $g^* \in G^*$, and $h^* \in H^*$ it can be shown that:

$$d(i,j) = \tfrac{1}{4}\Big(d(a^*,g^*) + d(a^*,h^*) + d(b^*,g^*) + d(b^*,h^*) - 2d(a^*,b^*) - 2d(g^*,h^*)\Big) \quad (4)$$

Empirically, one can obtain a more robust empirical estimate $\widehat{d}(i,j)$ by averaging over all valid choices of a^*, b^* in Eq. 3 and all valid choices of a^*, b^*, g^*, h^* in Eq. 4 (Desper and Gascuel, 2005).

3.2 Constructing a Spectral Additive Metric

In constructing our distance metric, we begin with the following assumption on the distribution in Eq. 1 (analogous to the assumptions made in Anandkumar et al., 2011).

Assumption 1 (Linear, Rank m, Means)

$$\mathbb{E}[z_i|\pi_{\boldsymbol{x}}(z_i), \boldsymbol{x}] = \boldsymbol{A}_{(z_i|z_{\pi_{\boldsymbol{x}}(z_i)},\boldsymbol{x})}\pi_{\boldsymbol{x}}(z_i)\ \forall i \in [H]$$

where $\boldsymbol{A}_{(z_i|\pi_{\boldsymbol{x}}(z_i),\boldsymbol{x})} \in \mathbb{R}^{m\times m}$ *has rank* m.

$$\mathbb{E}[w_i|\pi_{\boldsymbol{x}}(w_i), \boldsymbol{x}] = \boldsymbol{C}_{(w_i|\pi_{\boldsymbol{x}}(w_i),\boldsymbol{x})}\pi_{\boldsymbol{x}}(w_i)\ \forall i \in [\ell(\boldsymbol{x})]$$

where $\boldsymbol{C}_{(w_i|\pi_{\boldsymbol{x}}(w_i),\boldsymbol{x})} \in \mathbb{R}^{p\times m}$ *has rank* m.

Also assume that $\mathbb{E}[z_i z_i^\top|\boldsymbol{x}]$ *has rank* $m\ \forall i \in [H]$.

Note that the matrices $\boldsymbol{A}$ and $\boldsymbol{C}$ are a direct function of $\theta(\boldsymbol{x})$, but we do not specify a model family for $\theta(\boldsymbol{x})$. The only restriction is in the form of the above assumption. If w_i and z_i were discrete, represented as binary vectors, the above assumption would correspond to requiring all conditional probability tables in the latent tree to have rank m. Assumption 1 allows for the w_i to be high dimensional features, as long as the expectation requirement above is satisfied. Similar assumptions are made with spectral parameter learning methods e.g. Hsu et al. (2009), Bailly et al. (2009), Parikh et al. (2011), and Cohen et al. (2012).

Furthermore, Assumption 1 makes it explicit that regardless of the size of p, the relationships among the variables in the latent tree are restricted to be of rank m, and are thus *low rank* since $p > m$. To leverage this low rank structure, we propose using the following additive metric, a normalized variant of that in Anandkumar et al. (2011):

$$d^{\text{spectral}}(i,j) = -\log\Lambda_m(\boldsymbol{\Sigma}_{\boldsymbol{x}}(i,j)) + \tfrac{1}{2}\log\Lambda_m(\boldsymbol{\Sigma}_{\boldsymbol{x}}(i,i)) + \tfrac{1}{2}\log\Lambda_m(\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,j)) \quad (5)$$

where $\Lambda_m(\boldsymbol{A})$ denotes the product of the top m singular values of $\boldsymbol{A}$ and $\boldsymbol{\Sigma}_{\boldsymbol{x}}(i,j) := \mathbb{E}[v_i v_j^\top|\boldsymbol{x}]$, i.e. the uncentered cross-covariance matrix.

We can then show that this metric is additive:

Lemma 1 *If Assumption 1 holds then,* d^{spectral} *is an additive tree metric (Definition 1).*

A proof is in the supplementary for completeness. From here, we use d to denote d^{spectral}, since that is the metric we use for our learning algorithm.

3.3 Recovering the Minimal Projective Latent Tree

It has been shown (Rzhetsky and Nei, 1993) that for any additive tree metric, $u(\boldsymbol{x})$ can be recovered by solving $\arg\min_{u \in \mathcal{U}} c(u)$ for $c(u)$:

$$c(u) = \sum_{(i,j) \in \mathcal{E}_u} d(i,j). \quad (6)$$

where $\mathcal{E}_u$ is the set of pairs of nodes which are adjacent to each other in u and $d(i,j)$ is computed using Eq. 3 and Eq. 4.

Note that the metric d we use in defining $c(u)$ is based on the expectations from the true distribution. In practice, the true distribution is unknown, and therefore we use an approximation for the distance metric $\hat{d}$. As we discussed in §3.1 all elements of the distance matrix are functions of observable quantities if the underlying tree u is known. However, only the word-word sub-block $\boldsymbol{D}_{WW}$ can be directly estimated from the data without knowledge of the tree structure.

This subtlety makes solving the minimization problem in Eq. 6 NP-hard (Desper and Gascuel, 2005) if u is allowed to be an arbitrary undirected tree. However, if we restrict u to be in $\mathcal{U}$, as we do in the above, then maximizing $\hat{c}(u)$ over $\mathcal{U}$ can be solved using the bilexical parsing algorithm from Eisner and Satta (1999). This is because the computation of the other sub-blocks of the distance matrix only depend on the partitions of the nodes shown in Figure 3 into A, B, G, and H, and not on the entire tree structure.

Therefore, the procedure to find a bracketing for a given POS tag $\boldsymbol{x}$ is to first estimate the distance matrix sub-block $\widehat{\boldsymbol{D}}_{WW}$ from raw text data (see §3.4), and then solve the optimization problem $\arg\min_{u \in \mathcal{U}} \hat{c}(u)$ using a variant of the Eisner-Satta algorithm where $\hat{c}(u)$ is identical to $c(u)$ in Eq. 6, with d replaced with $\hat{d}$.

Summary. We first defined a generative model that describes how a sentence, its sequence of POS tags, and its bracketing is generated (§2.3). First an undirected $u \in \mathcal{U}$ is generated (only as a function of the POS tags), and then u is mapped to a bracketing using a direction mapping h_{dir}. We then showed that we can define a distance metric between nodes in the undirected tree, such that minimizing it leads to a recovery of u. This distance metric can be computed based only on the text, without needing to identify the latent information (§3.2). If the true distance metric is known, with respect to the true distribution that generates the words in a sentence, then u can be fully recovered by optimizing the cost function $c(u)$. However, in practice the distance metric must be estimated from data, as discussed below.

Algorithm 1 The learning algorithm for finding the latent structure from a set of examples $(\boldsymbol{w}^{(i)}, \boldsymbol{x}^{(i)})$, $i \in [N]$.

Inputs: Set of examples $(\boldsymbol{w}^{(i)}, \boldsymbol{x}^{(i)})$ for $i \in [N]$, a kernel $K_\gamma(j,k,j',k'|\boldsymbol{x},\boldsymbol{x}')$, an integer m

Data structures: For each $i \in [N], j,k \in \ell(\boldsymbol{x}^{(i)})$ there is a (uncentered) covariance matrix $\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{x}^{(i)}}(j,k) \in \mathbb{R}^{p \times p}$, and a distance $\hat{d}^{\text{spectral}}(j,k)$.

Algorithm:

(Covariance estimation) $\forall i \in [N], j,k \in \ell(\boldsymbol{x}^{(i)})$

- Let $C_{j',k'|i'} = w^{(i')}_{j'} (w^{(i')}_{k'})^\top$, $k_{j,k,j',k',i,i'} = K_\gamma(j,k,j',k'|\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(i')})$ and $\ell_{i'} = \ell(\boldsymbol{x}^{(i')})$, and estimate each $p \times p$ covariance matrix as:

$$\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{x}}(j,k) = \frac{\sum_{i'=1}^{N} \sum_{j'=1}^{\ell_{i'}} \sum_{k'=1}^{\ell_{i'}} k_{j,k,j',k',i,i'} C_{j',k'|i'}}{\sum_{i'=1}^{N} \sum_{j'=1}^{\ell_{i'}} \sum_{k'=1}^{\ell_{i'}} k_{j,k,j',k',i,i'}}$$

- Compute $\hat{d}^{\text{spectral}}(j,k)$ $\forall j,k \in \ell(\boldsymbol{x}^{(i)})$ using Eq. 5.

(Uncover structure) $\forall i \in [N]$

- Find $\hat{u}^{(i)} = \arg\min_{u \in \mathcal{U}} \hat{c}(u)$, and for the ith example, return the structure $h_{\text{dir}}(\hat{u}^{(i)})$.

3.4 Estimation of d from Sparse Data

We now address the data sparsity problem, in particular that $\mathcal{D}(\boldsymbol{x})$ can be very small, and therefore estimating d for each POS sequence separately can be problematic.[3]

In order to estimate d from data, we need to estimate the covariance matrices $\boldsymbol{\Sigma}_{\boldsymbol{x}}(i,j)$ (for $i,j \in \{1,\ldots,\ell(\boldsymbol{x})\}$) from Eq. 5.

To give some motivation to our solution, consider estimating the covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{x}}(1,2)$ for the tag sequence $\boldsymbol{x} = (\texttt{DT}_1, \texttt{NN}_2, \texttt{VBD}_3, \texttt{DT}_4, \texttt{NN}_5)$. $\mathcal{D}(\boldsymbol{x})$ may be insufficient for an accurate empirical es-

[3] This data sparsity problem is quite severe – for example, the Penn treebank (Marcus et al., 1993) has a total number of 43,498 sentences, with 42,246 *unique* POS tag sequences, averaging $|\mathcal{D}(\boldsymbol{x})|$ to be 1.04.

timate. However, consider another sequence $\boldsymbol{x}' = (\texttt{RB}_1, \texttt{DT}_2, \texttt{NN}_3, \texttt{VBD}_4, \texttt{DT}_5, \texttt{ADJ}_6, \texttt{NN}_7)$. Although $\boldsymbol{x}$ and $\boldsymbol{x}'$ are not identical, it is likely that $\boldsymbol{\Sigma}_{\boldsymbol{x}'}(2,3)$ is similar to $\boldsymbol{\Sigma}_{\boldsymbol{x}}(1,2)$ because the determiner and the noun appear in similar syntactic context. $\boldsymbol{\Sigma}_{\boldsymbol{x}'}(5,7)$ also may be somewhat similar, but $\boldsymbol{\Sigma}_{\boldsymbol{x}'}(2,7)$ should not be very similar to $\boldsymbol{\Sigma}_{\boldsymbol{x}}(1,2)$ because the noun and the determiner appear in a different syntactic context.

The observation that the covariance matrices depend on local syntactic context is the main driving force behind our solution. The local syntactic context acts as an "anchor," which enhances or replaces a word index in a sentence with local syntactic context. More formally, an anchor is a function G that maps a word index j and a sequence of POS tags $\boldsymbol{x}$ to a local context $G(j, \boldsymbol{x})$. The anchor we use is $G(j, \boldsymbol{x}) = (j, x_j)$. Then, the covariance matrices $\boldsymbol{\Sigma}_{\boldsymbol{x}}$ are estimated using kernel smoothing (Hastie et al., 2009), where the smoother tests similarity between the different anchors $G(j, \boldsymbol{x})$.

The full learning algorithm is given in Figure 1. The first step in the algorithm is to estimate the covariance matrix block $\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{x}^{(i)}}(j,k)$ for each training example $\boldsymbol{x}^{(i)}$ and each pair of preterminal positions (j,k) in $\boldsymbol{x}^{(i)}$. Instead of computing this block by computing the empirical covariance matrix for positions (j,k) in the data $\mathcal{D}(\boldsymbol{x})$, the algorithm uses all of the pairs (j',k') from all of N training examples. It averages the empirical covariance matrices from these contexts using a kernel weight, which gives a similarity measure for the position (j,k) in $\boldsymbol{x}^{(i)}$ and (j',k') in another example $\boldsymbol{x}^{(i')}$. γ is the kernel "bandwidth", a user-specified parameter that controls how inclusive the kernel will be with respect to examples in $\mathcal{D}$ (see § 4.1 for a concrete example). Note that the learning algorithm is such that it ensures that $\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{x}^{(i)}}(j,k) = \widehat{\boldsymbol{\Sigma}}_{\boldsymbol{x}^{(i')}}(j',k')$ if $G(j, \boldsymbol{x}^{(i)}) = G(j', \boldsymbol{x}^{(i')})$ and $G(k, \boldsymbol{x}^{(i)}) = G(k', \boldsymbol{x}^{(i')})$.

Once the empirical estimates for the covariance matrices are obtained, a variant of the Eisner-Satta algorithm is used, as mentioned in §3.3.

3.5 Theoretical Guarantees

Our main theoretical guarantee is that Algorithm 1 will recover the correct tree $u \in \mathcal{U}$ with high probability, if the given top bracket is correct and if we obtain enough examples $(\boldsymbol{w}^{(i)}, \boldsymbol{x}^{(i)})$ from the model in §2. We give the theorem statement below. The constants lurking in the O-notation and the full proof are in the supplementary.

Denote $\sigma_{\boldsymbol{x}}(j,k)^{(r)}$ as the r^{th} singular value of $\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,k)$. Let $\sigma^*(x) := \min_{j,k \in \ell(\boldsymbol{x})} \min\left(\sigma_{\boldsymbol{x}}(j,k)^{(m)}\right)$.

Theorem 1 *Define $\hat{u}$ as the estimated tree for tag sequence $\boldsymbol{x}$ and $u(\boldsymbol{x})$ as the correct tree. Let*

$$\triangle(\boldsymbol{x}) := \min_{u' \in \mathcal{U}: u' \neq u(\boldsymbol{x})} (c(u(\boldsymbol{x})) - c(u'))/(8|\ell(\boldsymbol{x})|)$$

Assume that

$$N \geq \mathcal{O}\left(\frac{m^2 \log\left(\frac{p^2 \ell(\boldsymbol{x})^2}{\delta}\right)}{\min(\sigma^*(\boldsymbol{x})^2 \triangle(\boldsymbol{x})^2, \sigma^*(\boldsymbol{x})^2)\nu_{\boldsymbol{x}}(\gamma)^2}\right)$$

Then with probability $1-\delta$, $\hat{u} = u(\boldsymbol{x})$.

where $\nu_{\boldsymbol{x}}(\gamma)$, defined in the supplementary, is a function of the underlying distribution over the tag sequences $\boldsymbol{x}$ and the kernel bandwidth γ.

Thus, the sample complexity of our approach depends on the dimensionality of the latent and observed states (m and p), the underlying singular values of the cross-covariance matrices ($\sigma^*(\boldsymbol{x})$) and the difference in the cost of the true tree compared to the cost of the incorrect trees ($\triangle(\boldsymbol{x})$).

4 Experiments

We report results on three different languages: English, German, and Chinese. For English we use the Penn treebank (Marcus et al., 1993), with sections 2–21 for training and section 23 for final testing. For German and Chinese we use the Negra treebank and the Chinese treebank respectively and the first 80% of the sentences are used for training and the last 20% for testing. All punctuation from the data is removed.[4]

We primarily compare our method to the constituent-context model (CCM) of Klein and Manning (2002). We also compare our method to the algorithm of Seginer (2007).

4.1 Experimental Settings

Top bracket heuristic Our algorithm requires the top bracket in order to direct the latent tree. In practice, we employ the following heuristic to find the bracket using the following three steps:

- If there exists a comma/semicolon/colon at index i that has at least a verb before i and both a noun followed by a verb after i, then return $([0, i-1], [i, \ell(x)])$ as the top bracket. (Pick the rightmost comma/semicolon/colon if multiple satisfy the criterion).

[4]We make brief use of punctuation for our top bracket heuristic detailed below before removing it.

Length	CCM	CCM-U	CCM-OB	CCM-UB
≤ 10	72.5	57.1	58.2	62.9
≤ 15	54.1	36	24	23.7
≤ 20	50	34.7	19.3	19.1
≤ 25	47.2	30.7	16.8	16.6
≤ 30	44.8	29.6	15.3	15.2
≤ 40	26.3	13.5	13.9	13.8

Table 1: Comparison of different CCM variants on English (training). U stands for universal POS tagset, OB stands for conjoining original POS tags with Brown clusters and UB stands for conjoining universal POS tags with Brown clusters. The best setting is just the vanilla setting, CCM.

- Otherwise find the first non-participle verb (say at index j) and return $([0, j-1], [j, \ell(\boldsymbol{x})])$.
- If no verb exists, return $([0, 1], [1, \ell(\boldsymbol{x})])$.

Word embeddings As mentioned earlier, each w_i can be an arbitrary feature vector. For all languages we use Brown clustering (Brown et al., 1992) to construct a $\log(C) + C$ feature vector where the first $\log(C)$ elements indicate which mergable cluster the word belongs to, and the last C elements indicate the cluster identity. For English, more sophisticated word embeddings are easily obtainable, and we experiment with neural word embeddings Turian et al. (2010) of length 50. We also explored two types of CCA embeddings: OSCCA and TSCCA, given in Dhillon et al. (2012). The OSCCA embeddings behaved better, so we only report its results.

Choice of kernel For our experiments, we use the kernel

$$K_\gamma(j,k,j',k'|\boldsymbol{x},\boldsymbol{x}') = \max\left\{0, 1 - \frac{\kappa(j,k,j',k'|\boldsymbol{x},\boldsymbol{x}')}{\gamma}\right\}$$

where γ denotes the user-specified bandwidth, and $\kappa(j,k,j',k'|\boldsymbol{x},\boldsymbol{x}') = \dfrac{|j-k|-|j'-k'|}{|j-k|+|j'-k'|}$ if $\boldsymbol{x}(j) = \boldsymbol{x}(j')$ and $\boldsymbol{x}(k') = \boldsymbol{x}(k)$, and $\text{sign}(j-k) = \text{sign}(j'-k')$ (and ∞ otherwise).

The kernel is non-zero if and only if the tags at position j and k in $\boldsymbol{x}$ are identical to the ones in position j' and k' in $\boldsymbol{x}'$, and if the direction between j and k is identical to the one between j' and k'. Note that the kernel is not binary, as opposed to the theoretical kernel in the supplementary material. Our experiments show that using a non-zero value different than 1 that is a function of the distance between j and k compared to the distance between j' and k' does better in practice.

Choice of data For CCM, we found that if the full dataset (all sentence lengths) is used in training, then performance degrades when evaluating on sentences of length ≤ 10. We therefore restrict the data used with CCM to sentences of length $\leq \ell$, where ℓ is the maximal sentence length being evaluated. This does not happen with our algorithm, which manages to leverage lexical information whenever more data is available. We therefore use the full data for our method for all lengths.

We also experimented with the original POS tags and the universal POS tags of Petrov et al. (2011). Here, we found out that our method does better with the universal part of speech tags. For CCM, we also experimented with the original parts of speech, universal tags (CCM-U), the cross-product of the original parts of speech with the Brown clusters (CCM-OB), and the cross-product of the universal tags with the Brown clusters (CCM-UB). The results in Table 1 indicate that the vanilla setting is the best for CCM.

Thus, for all results, we use universal tags for our method and the original POS tags for CCM. We believe that our approach substitutes the need for fine-grained POS tags with the lexical information. CCM, on the other hand, is fully unlexicalized.

Parameter Selection Our method requires two parameters, the latent dimension m and the bandwidth γ. CCM also has two parameters, the number of extra constituent/distituent counts used for smoothing. For both methods we chose the best parameters for sentences of length $\ell \leq 10$ on the English Penn Treebank (training) and used this set for all other experiments. This resulted in $m = 7, \gamma = 0.4$ for our method and 2, 8 for CCM's extra constituent/distituent counts respectively. We also tried letting CCM choose different hyperparameters for different sentence lengths based on dev-set likelihood, but this gave worse results than holding them fixed.

4.2 Results

Test I: Accuracy Table 2 summarizes our results. CCM is used with the initializer proposed in Klein and Manning (2002).[5] NN, CC, and BC indicate the performance of our method for neural embeddings, CCA embeddings, and Brown clustering respectively, using the heuristic for h_{dir} de-

[5]We used the implementation available at http://tinyurl.com/lhwk5n6.

	ℓ	English							German			Chinese		
		NN-O	NN	CC-O	CC	BC-O	BC	CCM	BC-O	BC	CCM	BC-O	BC	CCM
train	≤ 10	70.9	69.2	70.4	68.7	71.1	69.3	72.5	64.6	59.9	62.6	64.9	57.3	46.1
	≤ 20	55.1	53.5	53.2	51.6	53.0	51.5	50	52.7	48.7	47.9	51.4	46	22.4
	≤ 40	46.1	44.5	43.6	41.9	43.3	41.8	26.3	46.7	43.6	19.8	42.6	38.6	15
test	≤ 10	69.2	66.7	68.3	65.5	68.9	66.1	70.5	66.4	61.6	64.7	58.0	53.2	40.7
	≤ 15	60.3	58.3	58.6	56.4	58.6	56.5	53.8	57.5	53.5	49.6	54.3	49.4	35.9
	≤ 20	54.1	52.3	52.3	50.3	51.9	50.2	50.4	52.8	49.2	48.9	49.7	45.5	20.1
	≤ 25	50.8	49.0	48.6	46.6	48.3	46.6	47.4	50.0	46.8	45.6	46.7	42.7	17.8
	≤ 30	48.1	46.3	45.6	43.7	45.4	43.8	44.9	48.3	45.4	21.9	44.6	40.7	16.1
	≤ 40	45.5	43.8	43.0	41.1	42.7	41.1	26.1	46.9	44.1	20.1	42.2	38.6	14.3

Table 2: F_1 bracketing measure for the test sets and train sets in three languages. NN, CC, and BC indicate the performance of our method for neural embeddings, CCA embeddings, and Brown clustering respectively, using the heuristic for h_{dir} described in § 4.1. NN-O, CC-O, and BC-O indicate that the oracle (i.e. true top bracket) was used for h_{dir}.

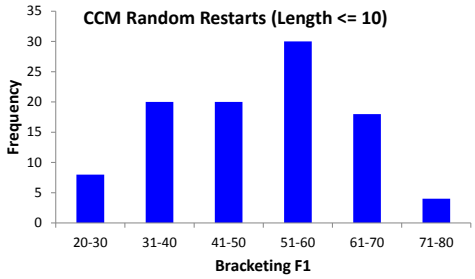


Figure 4: Histogram showing performance of CCM across 100 random restarts for sentences of length ≤ 10.

scribed in § 4.1. NN-O, CC-O, and BC-O indicate that the oracle (i.e. true top bracket) was used for h_{dir}. For our method, test set results can be obtained by using Algorithm 1 (except the distances are computed using the training data).

For English, while CCM behaves better for short sentences ($\ell \leq 10$), our algorithm is more robust with longer sentences. This is especially noticeable for length ≤ 40, where CCM breaks down and our algorithm is more stable. We find that the neural embeddings modestly outperform the CCA and Brown cluster embeddings.

The results for German are similar, except CCM breaks down earlier at sentences of $\ell \leq 30$. For Chinese, our method substantially outperforms CCM for all lengths. Note that CCM performs very poorly, obtaining only around 20% accuracy even for sentences of $\ell \leq 20$. We didn't have neural embeddings for German and Chinese (which worked best for English) and thus only used Brown cluster embeddings.

For English, the disparity between NN-O (oracle top bracket) and NN (heuristic top bracket) is rather low suggesting that our top bracket heuristic is rather effective. However, for German and Chinese note that the "BC-O" performs substantially better, suggesting that if we had a better top bracket heuristic our performance would increase.

Test II: Sensitivity to initialization The EM algorithm with the CCM requires very careful initialization, which is described in Klein and Manning (2002). If, on the other hand, random initialization is used, the variance of the performance of the CCM varies greatly. Figure 4 shows a histogram of the performance level for sentences of length ≤ 10 for different random initializers. As one can see, for some restarts, CCM obtains accuracies lower than 30% due to local optima. Our method does not suffer from local optima and thus does not require careful initialization.

Test III: Comparison to Seginer's algorithm Our approach is not directly comparable to Seginer's because he uses punctuation, while we use POS tags. Using Seginer's parser we were able to get results on the training sets. On English: 75.2% ($\ell \leq 10$), 64.2% ($\ell \leq 20$), 56.7% ($\ell \leq 40$). On German: 57.8% ($\ell \leq 10$), 45.0% ($\ell \leq 20$), and 39.9% ($\ell \leq 40$). On Chinese: 56.6% ($\ell \leq 10$), 45.1% ($\ell \leq 20$), and 38.9% ($\ell \leq 40$).

Thus, while Seginer's method performs better on English, our approach performs 2-3 points better on German, and both methods give similar performance on Chinese.

5 Conclusion

We described a spectral approach for unsupervised constituent parsing that comes with theoretical guarantees on latent structure recovery. Empirically, our algorithm performs favorably to the CCM of Klein and Manning (2002) without the need for careful initialization.

Acknowledgements: This work is supported by NSF IIS1218282, NSF IIS1111142, NIH R01GM093156, and the NSF Graduate Research Fellowship Program under Grant No. 0946825 (NSF Fellowship to APP).

References

A. Anandkumar, K. Chaudhuri, D. Hsu, S. M. Kakade, L. Song, and T. Zhang. 2011. Spectral methods for learning multivariate latent tree structure. *arXiv preprint arXiv:1107.1283*.

R. Bailly, F. Denis, and L. Ralaivola. 2009. Grammatical inference as a principal component analysis problem. In *Proceedings of ICML*.

R. Bailly, X. Carreras, F. M. Luque, and A. Quattoni. 2013. Unsupervised spectral learning of WCFG as low-rank matrix completion. In *Proceedings of EMNLP*.

P. F. Brown, P.V. Desouza, R.L. Mercer, V.J.D. Pietra, and J.C. Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

O. P. Buneman. 1971. The recovery of trees from measures of dissimilarity. *Mathematics in the archaeological and historical sciences*.

P. Buneman. 1974. A note on the metric properties of trees. *Journal of Combinatorial Theory, Series B*, 17(1):48–50.

M.J. Choi, V. YF Tan, A. Anandkumar, and A.S. Willsky. 2011. Learning latent tree graphical models. *The Journal of Machine Learning Research*, 12:1771–1812.

C. K. Chow and C. N. Liu. 1968. Approximating Discrete Probability Distributions With Dependence Trees. *IEEE Transactions on Information Theory*, IT-14:462–467.

S. B. Cohen and N. A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of HLT-NAACL*.

S. B. Cohen and N. A. Smith. 2012. Empirical risk minimization for probabilistic grammars: Sample complexity and hardness of learning. *Computational Linguistics*, 38(3):479–526.

S. B. Cohen, K. Stratos, M. Collins, D. P. Foster, and L. Ungar. 2012. Spectral learning of latent-variable PCFGs. In *Proceedings of ACL*.

R. Desper and O. Gascuel. 2005. The minimum evolution distance-based approach to phylogenetic inference. *Mathematics of evolution and phylogeny*, pages 1–32.

P. S. Dhillon, J. Rodu, D. P. Foster, and L. H. Ungar. 2012. Two step cca: A new spectral method for estimating vector models of words. In *Proceedings of ICML*.

J. Eisner and G. Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of ACL*.

P. Erdõs, M. Steel, L. Székely, and T. Warnow. 1999. A few logs suffice to build (almost) all trees: Part ii. *Theoretical Computer Science*, 221(1):77–118.

J. Gillenwater, K. Ganchev, J. Graça, F. Pereira, and B. Taskar. 2010. Sparsity in dependency grammar induction. In *Proceedings of ACL*.

K. Gimpel and N.A. Smith. 2012. Concavity and initialization for unsupervised dependency parsing. In *Proceedings of NAACL*.

D. Golland, J. DeNero, and J. Uszkoreit. 2012. A feature-rich constituent context model for grammar induction. In *Proceedings of ACL*.

M. Gormley and J. Eisner. 2013. Nonconvex global optimization for latent-variable models. In *Proceedings of ACL*.

S. Harmeling and C. KI Williams. 2011. Greedy learning of binary latent trees. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(6):1087–1097.

T. Hastie, R. Tibshirani, and J. Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer Verlag.

W. P. Headden, M. Johnson, and D. McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of NAACL-HLT*.

D. Hsu, S. Kakade, and T. Zhang. 2009. A spectral algorithm for learning hidden Markov models. In *Proceedings of COLT*.

D. Hsu, S. M. Kakade, and P. Liang. 2012. Identifiability and unmixing of latent parse trees. *arXiv preprint arXiv:1206.3137*.

M. Ishteva, H. Park, and L. Song. 2012. Unfolding latent tree structures using 4th order tensors. *arXiv preprint arXiv:1210.1258*.

F. Jelinek, J. D. Lafferty, and R. L. Mercer. 1992. *Basic methods of probabilistic context free grammars*. Springer.

D. Klein and C. D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of ACL*.

M. Kolar, A. P. Parikh, and E. P. Xing. 2010a. On sparse nonparametric conditional covariance selection. In *Proceedings of ICML*.

M. Kolar, L. Song, A. Ahmed, and E. P. Xing. 2010b. Estimating time-varying networks. *The Annals of Applied Statistics*, 4(1):94–123.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19:313–330.

A.P. Parikh, L. Song, and E.P. Xing. 2011. A spectral algorithm for latent tree graphical models. In *Proceedings of ICML*.

S. Petrov, D. Das, and R. McDonald. 2011. A universal part-of-speech tagset. *ArXiv:1104.2086*.

A. Rzhetsky and M. Nei. 1993. Theoretical foundation of the minimum-evolution method of phylogenetic inference. *Molecular Biology and Evolution*, 10(5):1073–1095.

N. Saitou and M. Nei. 1987. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, 4(4):406–425.

Y. Seginer. 2007. Fast unsupervised incremental parsing. In *Proceedings of ACL*.

N. A. Smith and J. Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of ACL*.

L. Song, A.P. Parikh, and E.P. Xing. 2011. Kernel embeddings of latent tree graphical models. In *Proceedings of NIPS*.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. 2010a. From baby steps to leapfrog: how less is more in unsupervised dependency parsing. In *Proceedings of NAACL*.

V. I. Spitkovsky, H. Alshawi, D. Jurafsky, and C. D. Manning. 2010b. Viterbi training improves unsupervised dependency parsing. In *Proceedings of CoNLL*.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. 2013. Breaking out of local optima with count transforms and model recombination: A study in grammar induction. In *Proceedings of EMNLP*.

J. P. Turian, L.-A. Ratinov, and Y. Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of ACL*.

S. Zhou, J. Lafferty, and L. Wasserman. 2010. Time varying undirected graphs. *Machine Learning*, 80(2-3):295–319.

Weak semantic context helps phonetic learning in a model of infant language acquisition

Stella Frank
sfrank@inf.ed.ac.uk
ILCC, School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK

Naomi H. Feldman
nhf@umd.edu
Department of Linguistics
University of Maryland
College Park, MD, 20742, USA

Sharon Goldwater
sgwater@inf.ed.ac.uk
ILCC, School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK

Abstract

Learning phonetic categories is one of the first steps to learning a language, yet is hard to do using only distributional phonetic information. Semantics could potentially be useful, since words with different meanings have distinct phonetics, but it is unclear how many word meanings are known to infants learning phonetic categories. We show that attending to a weaker source of semantics, in the form of a distribution over topics in the current context, can lead to improvements in phonetic category learning. In our model, an extension of a previous model of joint word-form and phonetic category inference, the probability of word-forms is topic-dependent, enabling the model to find significantly better phonetic vowel categories and word-forms than a model with no semantic knowledge.

1 Introduction

Infants begin learning the phonetic categories of their native language in their first year (Kuhl et al., 1992; Polka and Werker, 1994; Werker and Tees, 1984). In theory, semantic information could offer a valuable cue for phoneme induction[1] by helping infants distinguish between minimal pairs, as linguists do (Trubetzkoy, 1939). However, due to a widespread assumption that infants do not know the meanings of many words at the age when they are learning phonetic categories (see Swingley, 2009 for a review), most recent models of early phonetic category acquisition have explored the phonetic learning problem in the absence of semantic information (de Boer and Kuhl, 2003; Dillon et al., 2013; Feldman et al., 2013a; McMurray et al., 2009; Vallabha et al., 2007).

Models without any semantic information are likely to underestimate infants' ability to learn phonetic categories. Infants learn language in the wild, and quickly attune to the fact that words have (possibly unknown) meanings. The extent of infants' semantic knowledge is not yet known, but existing evidence shows that six-month-olds can associate some words with their referents (Bergelson and Swingley, 2012; Tincoff and Jusczyk, 1999, 2012), leverage non-acoustic contexts such as objects or articulations to distinguish similar sounds (Teinonen et al., 2008; Yeung and Werker, 2009), and map meaning (in the form of objects or images) to new word-forms in some laboratory settings (Friedrich and Friederici, 2011; Gogate and Bahrick, 2001; Shukla et al., 2011). These findings indicate that young infants are sensitive to co-occurrences between linguistic stimuli and at least some aspects of the world.

In this paper we explore the potential contribution of semantic information to phonetic learning by formalizing a model in which learners attend to the word-level context in which phones appear (as in the lexical-phonetic learning model of Feldman et al., 2013a) and also to the situations in which word-forms are used. The modeled situations consist of combinations of categories of salient activities or objects, similar to the activity contexts explored by Roy et al. (2012), e.g.,'getting dressed' or 'eating breakfast'. We assume that child learners are able to infer a representation of the situational context from their non-linguistic environment. However, in our simulations we approximate the environmental information by running a topic model (Blei et al., 2003) over a corpus of child-directed speech to infer a topic distribution for each situation. These topic distributions are then used as input to our model to represent situational contexts.

The situational information in our model is simi-

[1] The models in this paper do not distinguish between phonetic and phonemic categories, since they do not capture phonological processes (and there are also none present in our synthetic data). We thus use the terms interchangeably.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1073–1083,
Baltimore, Maryland, USA, June 23-25 2014.

lar to that assumed by theories of cross-situational word learning (Frank et al., 2009; Smith and Yu, 2008; Yu and Smith, 2007), but our model does not require learners to map individual words to their referents. Even in the absence of word-meaning mappings, situational information is potentially useful because similar-sounding words uttered in similar situations are more likely to be tokens of the same lexeme (containing the same phones) than similar-sounding words uttered in different situations.

In simulations of vowel learning, inspired by Vallabha et al. (2007) and Feldman et al. (2013a), we show a clear improvement over previous models in both phonetic and lexical (word-form) categorization when situational context is used as an additional source of information. This improvement is especially noticeable when the word-level context is providing less information, arguably the more realistic setting. These results demonstrate that relying on situational co-occurrence can improve phonetic learning, even if learners do not yet know the meanings of individual words.

2 Background and overview of models

Infants attend to distributional characteristics of their input (Maye et al., 2002, 2008), leading to the hypothesis that phonetic categories could be acquired on the basis of bottom-up distributional learning alone (de Boer and Kuhl, 2003; Vallabha et al., 2007; McMurray et al., 2009). However, this would require sound categories to be well separated, which often is not the case—for example, see Figure 1, which shows the English vowel space that is the focus of this paper.

Recent work has investigated whether infants could overcome such distributional ambiguity by incorporating top-down information, in particular, the fact that phones appear within words. At six months, infants begin to recognize word-forms such as their name and other frequently occurring words (Mandel et al., 1995; Jusczyk and Hohne, 1997), without necessarily linking a meaning to these forms. This "protolexicon" can help differentiate phonetic categories by adding word contexts in which certain sound categories appear (Swingley, 2009; Feldman et al., 2013b). To explore this idea further, Feldman et al. (2013a) implemented the Lexical-Distributional (LD) model, which jointly learns a set of phonetic vowel categories and a set of word-forms containing those categories. Simulations showed that the use of lexical context greatly improved phonetic learning.

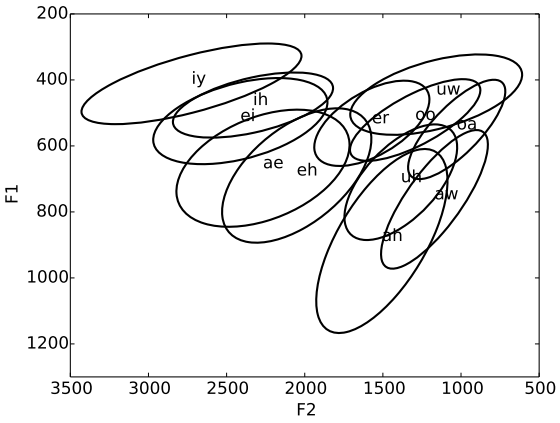


Figure 1: The English vowel space (generated from Hillenbrand et al. (1995), see Section 6.2), plotted using the first two formants.

Our own Topic-Lexical-Distributional (TLD) model extends the LD model to include an additional type of context: the situations in which words appear. To motivate this extension and clarify the differences between the models, we now provide a high-level overview of both models; details are given in Sections 3 and 4.

2.1 Overview of LD model

Both the LD and TLD models are computational-level models of phonetic (specifically, vowel) categorization where phones (vowels) are presented to the model in the context of words.[2] The task is to infer a set of phonetic categories and a set of lexical items on the basis of the data observed for each word token x_i. In the original LD model, the observations for token x_i are its frame f_i, which consists of a list of consonants and slots for vowels, and the list of vowel tokens $\boldsymbol{w}_i$. (The TLD model includes additional observations, described below.) A single vowel token, w_{ij}, is a two dimensional vector representing the first two formants (peaks in the frequency spectrum, ordered from lowest to highest). For example, a token of the word *kitty* would have the frame $f_i = \texttt{k\_t\_}$, containing two consonant phones, /k/ and /t/, with two vowel phone slots in between, and two vowel formant vectors,

[2]For a related model that also tackles the word segmentation problem, see Elsner et al. (2013). In a model of phonological learning, Fourtassi and Dupoux (submitted) show that semantic context information similar to that used here remains useful despite segmentation errors.

$w_{i0} = [464, 2294]$ and $w_{i1} = [412, 2760]$.[3]

Given the data, the model must assign each vowel token to a vowel category, $w_{ij} = c$. Both the LD and the TLD models do this using intermediate lexemes, $\boldsymbol{\ell}$, which contain vowel category assignments, $v_{\ell j} = c$, as well as a frame f_ℓ. If a word token is assigned to a lexeme, $x_i = \ell$, the vowels within the word are assigned to that lexeme's vowel categories, $w_{ij} = v_{\ell j} = c$.[4] The word and lexeme frames must match, $f_i = f_\ell$.

Lexical information helps with phonetic categorization because it can disambiguate highly overlapping categories, such as the *ae* and *eh* categories in Figure 1. A purely distributional learner who observes a cluster of data points in the *ae-eh* region is likely to assume all these points belong to a single category because the distributions of the categories are so similar. However, a learner who attends to lexical context will notice a difference: contexts that only occur with *ae* will be observed in one part of the *ae-eh* region, while contexts that only occur with *eh* will be observed in a different (though partially overlapping) space. The learner then has evidence of two different categories occurring in different sets of lexemes.

Simulations with the LD model show that using lexical information to constrain phonetic learning can greatly improve categorization accuracy (Feldman et al., 2013a), but it can also introduce errors. When two word tokens contain the same consonant frame but different vowels (i.e., minimal pairs), the model is more likely to categorize those two vowels together. Thus, the model has trouble distinguishing minimal pairs. Although young children also have trouble with minimal pairs (Stager and Werker, 1997; Thiessen, 2007), the LD model may overestimate the degree of the problem. We hypothesize that if a learner is able to associate words with the contexts of their use (as children likely are), this could provide a weak source of information for disambiguating minimal pairs even without knowing their exact meanings. That is, if the learner hears kV_1t and kV_2t in different situational contexts, they are likely to be different lexical items (and V_1 and V_2 different phones), despite the lexical similarity between them.

[3] In simulations we also experiment with frames in which consonants are not represented perfectly.

[4] The notation is overloaded: w_{ij} refers both to the vowel formants and the vowel category assignments, and x_i refers to both the token identity and its assignment to a lexeme.

2.2 Overview of TLD model

To demonstrate the benefit of situational information, we develop the Topic-Lexical-Distributional (TLD) model, which extends the LD model by assuming that words appear in *situations* analogous to documents in a topic model. Each situation h is associated with a mixture of topics θ_h, which is assumed to be observed. Thus, for the ith token in situation h, denoted x_{hi}, the observed data will be its frame f_{hi}, vowels $\boldsymbol{w}_{hi}$, and topic vector θ_h.

From an acquisition perspective, the observed topic distribution represents the child's knowledge of the context of the interaction: she can distinguish bathtime from dinnertime, and is able to recognize that some topics appear in certain contexts (e.g. animals on walks, vegetables at dinnertime) and not in others (few vegetables appear at bathtime). We assume that the child would learn these topics from observing the world around her and the co-occurrences of entities and activities in the world. Within any given situation, there might be a mixture of different (actual or possible) topics that are salient to the child. We assume further that as the child learns the language, she will begin to associate specific words with each topic as well.

Thus, in the TLD model, the words used in a situation are topic-dependent, implying meaning, but without pinpointing specific referents. Although the model observes the distribution of topics in each situation (corresponding to the child observing her non-linguistic environment), it must learn to associate each (phonetically and lexically ambiguous) word token with a particular topic from that distribution. The occurrence of similar-sounding words in different situations with mostly non-overlapping topics will provide evidence that those words belong to different topics and that they are therefore different lexemes. Conversely, potential minimal pairs that occur in situations with similar topic distributions are more likely to belong to the same topic and thus the same lexeme.

Although we assume that children infer topic distributions from the non-linguistic environment, we will use transcripts from CHILDES to create the word/phone learning input for our model. These transcripts are not annotated with environmental context, but Roy et al. (2012) found that topics learned from similar transcript data using a topic model were strongly correlated with immediate activities and contexts. We therefore obtain the topic distributions used as input to the TLD model by

training an LDA topic model (Blei et al., 2003) on a superset of the child-directed transcript data we use for lexical-phonetic learning, dividing the transcripts into small sections (the 'documents' in LDA) that serve as our distinct situations $\boldsymbol{h}$. As noted above, the learned document-topic distributions $\boldsymbol{\theta}$ are treated as observed variables in the TLD model to represent the situational context. The topic-word distributions learned by LDA are discarded, since these are based on the (correct and unambiguous) words in the transcript, whereas the TLD model is presented with phonetically ambiguous versions of these word tokens and must learn to disambiguate them and associate them with topics.

3 Lexical-Distributional Model

In this section we describe more formally the generative process for the LD model (Feldman et al., 2013a), a joint Bayesian model over phonetic categories and a lexicon, before describing the TLD extension in the following section.

The set of phonetic categories and the lexicon are both modeled using non-parametric Dirichlet Process priors, which return a potentially infinite number of categories or lexemes. A DP is parametrized as $DP(\alpha, H)$, where α is a real-valued hyperparameter and H is a base distribution. H may be continuous, as when it generates phonetic categories in formant space, or discrete, as when it generates lexemes as a list of phonetic categories.

A draw from a DP, $G \sim DP(\alpha, H)$, returns a distribution over a set of draws from H, i.e., a discrete distribution over a set of categories or lexemes generated by H. In the mixture model setting, the category assignments are then generated from G, with the datapoints themselves generated by the corresponding components from H. If H is infinite, the support of the DP is likewise infinite. During inference, we marginalize over G.

3.1 Phonetic Categories: IGMM

Following previous models of vowel learning (de Boer and Kuhl, 2003; Vallabha et al., 2007; McMurray et al., 2009; Dillon et al., 2013) we assume that vowel tokens are drawn from a Gaussian mixture model. The Infinite Gaussian Mixture Model (IGMM) (Rasmussen, 2000) includes a DP prior, as described above, in which the base distribution H_C generates multivariate Gaussians drawn from a Normal Inverse-Wishart prior.[5] Each observation, a formant vector w_{ij}, is drawn from the Gaussian corresponding to its category assignment c_{ij}:

$$\mu_c, \Sigma_c \sim H_C = \mathit{NIW}(\mu_0, \Sigma_0, \nu_0) \quad (1)$$

$$G_C \sim DP(\alpha_c, H_C) \quad (2)$$

$$c_{ij} \sim G_C \quad (3)$$

$$w_{ij} | c_{ij} = c \sim N(\mu_c, \Sigma_c) \quad (4)$$

The above model generates a category assignment c_{ij} for each vowel token w_{ij}. This is the baseline IGMM model, which clusters vowel tokens using bottom-up distributional information only; the LD model adds top-down information by assigning categories in the lexicon, rather than on the token level.

3.2 Lexicon

In the LD model, vowel phones appear within words drawn from the lexicon. Each such lexeme is represented as a frame plus a list of vowel categories $\boldsymbol{v}_\ell$. Lexeme assignments for each token are drawn from a DP with a lexicon-generating base distribution H_L. The category for each vowel token in the word is determined by the lexeme; the formant values are drawn from the corresponding Gaussian as in the IGMM:

$$G_L \sim DP(\alpha_l, H_L) \quad (5)$$

$$x_i = \ell \sim G_L \quad (6)$$

$$w_{ij} | v_{\ell j} = c \sim N(\mu_c, \Sigma_c) \quad (7)$$

H_L generates lexemes by first drawing the number of phones from a geometric distribution and the number of consonant phones from a binomial distribution. The consonants are then generated from a DP with a uniform base distribution (but note they are fixed at inference time, i.e., are observed categorically), while the vowel phones $\boldsymbol{v}_\ell$ are generated by the IGMM DP above, $v_{\ell j} \sim G_C$.

Note that two draws from H_L may result in identical lexemes; these are nonetheless considered to be separate (homophone) lexemes.

4 Topic-Lexical-Distributional Model

The TLD model retains the IGMM vowel phone component, but extends the lexicon of the LD model by adding topic-specific lexicons, which capture the notion that lexeme probabilities are topic-dependent. Specifically, the TLD model replaces

[5] This compound distribution is equivalent to $\Sigma_c \sim \mathit{IW}(\Sigma_0, \nu_0)$, $\mu_c | \Sigma_c \sim N(\mu_0, \frac{\Sigma_c}{\nu_0})$

the Dirichlet Process lexicon with a Hierarchical Dirichlet Process (HDP; Teh (2006)). In the HDP lexicon, a top-level global lexicon is generated as in the LD model. Topic-specific lexicons are then drawn from the global lexicon, containing a subset of the global lexicon (but since the size of the global lexicon is unbounded, so are the topic-specific lexicons). These topic-specific lexicons are used to generate the tokens in a similar manner to the LD model. There are a fixed number of lower level topic-lexicons; these are matched to the number of topics in the LDA model used to infer the topic distributions (see Section 6.4).

More formally, the global lexicon is generated as a top-level DP: $G_L \sim DP(\alpha_l, H_L)$ (see Section 3.2; remember H_L includes draws from the IGMM over vowel categories). G_L is in turn used as the base distribution in the topic-level DPs, $G_k \sim DP(\alpha_k, G_L)$. In the Chinese Restaurant Franchise metaphor often used to describe HDPs, G_L is a global menu of dishes (lexemes). The topic-specific lexicons are restaurants, each with its own distribution over dishes; this distribution is defined by seating customers (word tokens) at *tables*, each of which serves a single dish from the menu: all tokens x at the same table t are assigned to the same lexeme ℓ_t. Inference (Section 5) is defined in terms of tables rather than lexemes; if multiple tables draw the same dish from G_L, tokens at these tables share a lexeme.

In the TLD model, tokens appear within situations, each of which has a distribution over topics θ_h. Each token x_{hi} has a co-indexed topic assignment variable, z_{hi}, drawn from θ_h, designating the topic-lexicon from which the table for x_{hi} is to be drawn. The formant values for w_{hij} are drawn in the same way as in the LD model, given the lexeme assignment at x_{hi}. This results in the following model, shown in Figure 2:

$$G_L \sim DP(\alpha_l, H_L) \quad (8)$$

$$G_k \sim DP(\alpha_k, G_L) \quad (9)$$

$$z_{hi} \sim Mult(\theta_h) \quad (10)$$

$$x_{hi} = t | z_{hi} = k \sim G_k \quad (11)$$

$$w_{hij} | x_{hi} = t, v_{\ell_t j} = c \sim N(\mu_c, \Sigma_c) \quad (12)$$

5 Inference: Gibbs Sampling

We use Gibbs sampling to infer three sets of variables in the TLD model: assignments to vowel categories in the lexemes, assignments of tokens to topics, and assignments of tokens to tables (from which the assignment to lexemes can be read off).

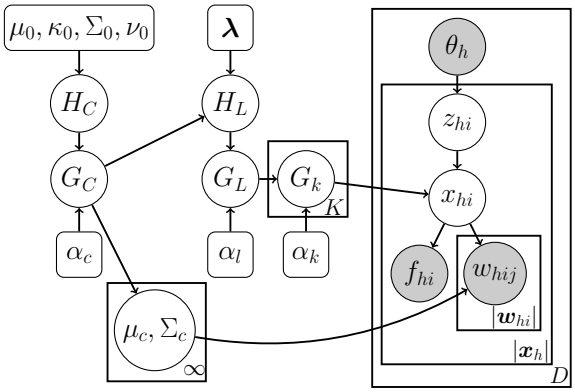


Figure 2: TLD model, depicting, from left to right, the IGMM component, the LD lexicon component, the topic-specific lexicons, and finally the token x_{hi}, appearing in document h, with observed vowel formants w_{hij} and frame f_{hi}. The lexeme assignment x_{hi} and the topic assignment z_{hi} are inferred, the latter using the observed document-topic distribution θ_h. Note that f_i is deterministic given the lexeme assignment. Squared nodes depict hyperparameters. $\boldsymbol{\lambda}$ is the set of hyperparameters used by H_L when generating lexical items (see Section 3.2).

5.1 Sampling lexeme vowel categories

Each vowel in the lexicon must be assigned to a category in the IGMM. The posterior probability of a category assignment is composed of the DP prior over categories and the likelihood of the observed vowels belonging to that category. We use $\boldsymbol{w}_{\ell j}$ to denote the set of vowel formants at position j in words that have been assigned to lexeme ℓ. Then,

$$P(v_{\ell j} = c | \boldsymbol{w}, \boldsymbol{x}, \boldsymbol{\ell}^{\setminus \ell}) \propto P(v_{\ell j} = c | \boldsymbol{\ell}^{\setminus \ell}) p(\boldsymbol{w}_{\ell j} | v_{\ell j} = c, \boldsymbol{w}^{\setminus \ell j}) \quad (13)$$

The first (DP prior) factor is defined as:

$$P(v_{\ell j} = c | \boldsymbol{v}^{\setminus \ell j}) = \begin{cases} \frac{n_c}{\sum_c n_c + \alpha_c} & \text{if } c \text{ exists} \\ \frac{\alpha_c}{\sum_c n_c + \alpha_c} & \text{if } c \text{ new} \end{cases} \quad (14)$$

where n_c is the number of other vowels in the lexicon, $\boldsymbol{v}^{\setminus lj}$, assigned to category c. Note that there is always positive probability of creating a new category.

The likelihood of the vowels is calculated by marginalizing over all possible means and variances of the Gaussian category parameters, given

the NIW prior. For a single point (if $|\boldsymbol{w}_{\ell j}| = 1$), this predictive posterior is in the form of a Student-t distribution; for the more general case see Feldman et al. (2013a), Eq. B3.

5.2 Sampling table & topic assignments

We jointly sample $\boldsymbol{x}$ and $\boldsymbol{z}$, the variables assigning tokens to tables and topics. Resampling the table assignment includes the possibility of changing to a table with a different lexeme or drawing a new table with a previously seen or novel lexeme. The joint conditional probability of a table and topic assignment, given all other current token assignments, is:

$$\begin{aligned} &P(x_{hi} = t, z_{hi} = k | \boldsymbol{w}_{hi}, \theta_h, \boldsymbol{t}^{\setminus hi}, \boldsymbol{\ell}, \boldsymbol{w}^{\setminus hi}) \\ &\quad = P(k|\theta_h) P(t|k, \ell_t, \boldsymbol{t}^{\setminus hi}) \\ &\qquad \prod_{c \in C} p(\boldsymbol{w}_{hi\cdot} | v_{\ell_t \cdot} = c, \boldsymbol{w}^{\setminus hi}) \end{aligned} \tag{15}$$

The first factor, the prior probability of topic k in document h, is given by θ_{hk} obtained from the LDA. The second factor is the prior probability of assigning word x_i to table t with lexeme ℓ given topic k. It is given by the HDP, and depends on whether the table t exists in the HDP topic-lexicon for k and, likewise, whether any table in the topic-lexicon has the lexeme ℓ:

$$P(t|k, \ell, \boldsymbol{t}^{\setminus hi}) \propto \begin{cases} \frac{n_{kt}}{n_k + \alpha_k} & \text{if } t \text{ in } k \\ \frac{\alpha_k}{n_k + \alpha_k} \frac{m_\ell}{m + \alpha_l} & \text{if } t \text{ new, } \ell \text{ known} \\ \frac{\alpha_k}{n_k + \alpha_k} \frac{\alpha_\ell}{m + \alpha_l} & \text{if } t \text{ and } \ell \text{ new} \end{cases} \tag{16}$$

Here n_{kt} is the number of other tokens at table t, n_k are the total number of tokens in topic k, m_ℓ is the number of tables across all topics with the lexeme ℓ, and m is the total number of tables.

The third factor, the likelihood of the vowel formants $\boldsymbol{w}_{hi}$ in the categories given by the lexeme $\boldsymbol{v}_l$, is of the same form as the likelihood of vowel categories when resampling lexeme vowel assignments. However, here it is calculated over the set of vowels in the token assigned to each vowel category (i.e., the vowels at indices where $v_{\ell_t \cdot} = c$). For a new lexeme, we approximate the likelihood using 100 samples drawn from the prior, each weighted by $\alpha/100$ (Neal, 2000).

5.3 Hyperparameters

The three hyperparameters governing the HDP over the lexicon, α_l and α_k, and the DP over vowel categories, α_c, are estimated using a slice sampler. The remaining hyperparameters for the vowel category and lexeme priors are set to the same values used by Feldman et al. (2013a).

6 Experiments

6.1 Corpus

We test our model on situated child directed speech, taken from the C1 section of the Brent corpus in CHILDES (Brent and Siskind, 2001; MacWhinney, 2000). This corpus consists of transcripts of speech directed at infants between the ages of 9 and 15 months, captured in a naturalistic setting as parent and child went about their day. This ensures variability of situations.

Utterances with unintelligible words or quotes are removed. We restrict the corpus to content words by retaining only words tagged as `adj`, `n`, `part` and `v` (adjectives, nouns, particles, and verbs). This is in line with evidence that infants distinguish content and function words on the basis of acoustic signals (Shi and Werker, 2003). Vowel categorization improves when attending only to more prosodically and phonologically salient tokens (Adriaans and Swingley, 2012), which generally appear within content, not function words. The final corpus consists of 13138 tokens and 1497 word types.

6.2 Hillenbrand Vowels

The transcripts do not include phonetic information, so, following Feldman et al. (2013a), we synthesize the formant values using data from Hillenbrand et al. (1995). This dataset consists of a set of 1669 manually gathered formant values from 139 American English speakers (men, women and children) for 12 vowels. For each vowel category, we construct a Gaussian from the mean and covariance of the datapoints belonging to that category, using the first and second formant values measured at steady state. We also construct a second dataset using only datapoints from adult female speakers.

Each word in the dataset is converted to a phonemic representation using the CMU pronunciation dictionary, which returns a sequence of Arpabet phoneme symbols. If there are multiple possible pronunciations, the first one is used. Each vowel phoneme in the word is then replaced by formant values drawn from the corresponding Hillenbrand Gaussian for that vowel.

6.3 Merging Consonant Categories

The Arpabet encoding used in the phonemic representation includes 24 consonants. We construct datasets both using the full set of consonants—the 'C24' dataset—and with less fine-grained consonant categories. Distinguishing all consonant categories assumes perfect learning of consonants prior to vowel categorization and is thus somewhat unrealistic (Polka and Werker, 1994), but provides an upper limit on the information that word-contexts can give.

In the 'C15' dataset, the voicing distinction is collapsed, leaving 15 consonant categories. The collapsed categories are B/P, G/K, D/T, CH/JH, V/F, TH/DH, S/Z, SH/ZH, R/L while HH, M, NG, N, W, Y remain separate phonemes. This dataset mirrors the finding in Mani and Plunkett (2010) that 12 month old infants are not sensitive to voicing mispronunciations.

The 'C6' dataset distinguishes between only 6 coarse consonant phonemes, corresponding to stops (B,P,G,K,D,T), affricates (CH,JH), fricatives (V, F, TH, DH, S, Z, SH, ZH, HH), nasals (M, NG, N), liquids (R, L), and semivowels/glides (W, Y). This dataset makes minimal assumptions about the category categories that infants could use in this learning setting.

Decreasing the number of consonants increases the ambiguity in the corpus: *bat* not only shares a frame (b_t) with *boat* and *bite*, but also, in the C15 dataset, with *put*, *pad* and *bad* (b/p_d/t), and in the C6 dataset, with *dog* and *kite*, among many others (STOP_STOP). Table 1 shows the percentage of types and tokens that are ambiguous in each dataset, that is, words in frames that match multiple wordtypes. Note that we always evaluate against the *gold* word identities, even when these are not distinguished in the model's input. These datasets are intended to evaluate the degree of reliance on consonant information in the LD and TLD models, and to what extent the topics in the TLD model can replace this information.

6.4 Topics

The input to the TLD model includes a distribution over topics for each situation, which we infer in advance from the full Brent corpus (not only the C1 subset) using LDA. Each transcript in the Brent corpus captures about 75 minutes of parent-child interaction, and thus multiple situations will be included in each file. The transcripts do not delimit situations, so we do this somewhat arbitrarily by splitting each transcript after 50 CDS utterances, resulting in 203 situations for the Brent C1 dataset. As well as function words, we also remove the five most frequent content words (*be, go, get, want, come*). On average, situations are only 59 words long, reflecting the relative lack of content words in CDS utterances.

Dataset	C24	C15	C6
Input Types	1487	1426	1203
Frames	1259	1078	702
Ambig Types %	27.2	42.0	80.4
Ambig Tokens %	41.3	56.9	77.2

Table 1: Corpus statistics showing the increasing amount of ambiguity as consonant categories are merged. Input types are the number of word types with distinct input representations (as opposed to gold orthographic word types, of which there are 1497). Ambiguous types and tokens are those with frames that match multiple (orthographic) word types.

We infer 50 topics for this set of situations using the `mallet` toolkit (McCallum, 2002). Hyperparameters are inferred, which leads to a dominant topic that includes mainly light verbs (*have, let, see, do*). The other topics are less frequent but capture stronger semantic meaning (e.g. *yummy, peach, cookie, daddy, bib* in one topic, *shoe, let, put, hat, pants* in another). The word-topic assignments are used to calculate unsmoothed situation-topic distributions $\boldsymbol{\theta}$ used by the TLD model.

6.5 Evaluation

We evaluate against adult categories, i.e., the 'gold-standard', since all learners of a language eventually converge on similar categories. (Since our model is not a model of the learning process, we do not compare the infant learning process to the learning algorithm.) We evaluate both the inferred phonetic categories and words using the clustering evaluation measure V-Measure (VM; Rosenberg and Hirschberg, 2007).[6] VM is the harmonic mean of two components, similar to F-score, where the components (VC and VH) are measures of cross entropy between the gold and model categorization.

[6]Other clustering measures, such as 1-1 matching and pairwise precision and recall (accuracy and completeness) showed the same trends, but VM has been demonstrated to be the most stable measure when comparing solutions with varying numbers of clusters (Christodoulopoulos et al., 2010).

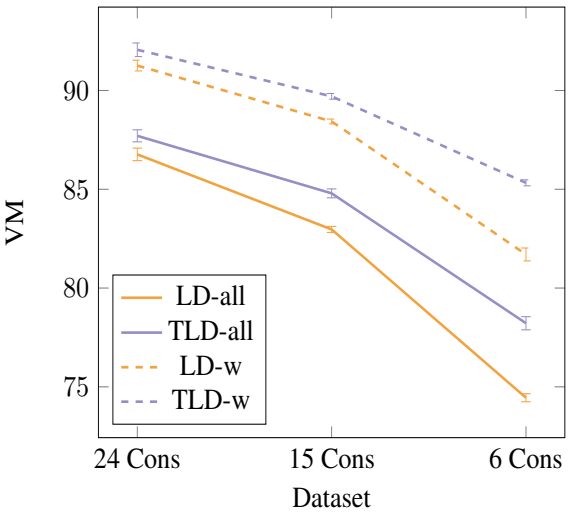


Figure 3: Vowel evaluation. 'all' refers to datasets with vowels synthesized from all speakers, 'w' to datasets with vowels synthesized from adult female speakers' vowels. The bars show a 95% Confidence Interval based on 5 runs. IGMM-all results in a VM score of 53.9 (CI=0.5); IGMM-w has a VM score of 65.0 (CI=0.2), not shown.

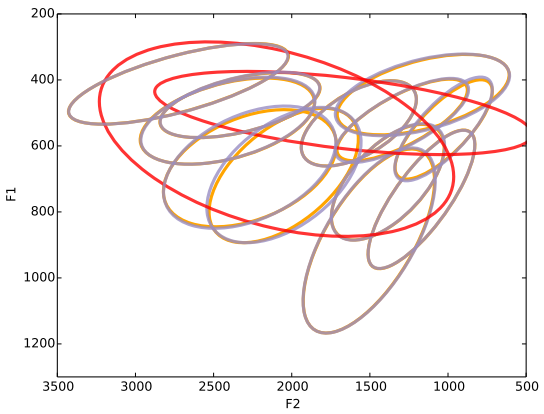


Figure 4: Vowels found by the TLD model; supervowels are indicated in red. The gold-standard vowels are shown in gold in the background but are mostly overlapped by the inferred categories.

For vowels, VM measures how well the inferred phonetic categorizations match the gold categories; for lexemes, it measures whether tokens have been assigned to the same lexemes both by the model and the gold standard. Words are evaluated against gold orthography, so homophones, e.g. *hole* and *whole*, are distinct gold words.

6.6 Results

We compare all three models—TLD, LD, and IGMM—on the vowel categorization task, and TLD and LD on the lexical categorization task (since IGMM does not infer a lexicon). The datasets correspond to two sets of conditions: firstly, either using vowel categories synthesized from all speakers or only adult female speakers, and secondly, varying the coarseness of the observed consonant categories. Each condition (model, vowel speakers, consonant set) is run five times, using 1500 iterations of Gibbs sampling with hyperparameter sampling. Overall, we find that TLD outperforms the other models in both tasks, across all conditions.

Vowel categorization results are shown in Figure 3. IGMM performs substantially worse than both TLD and LD, with scores more than 30 points lower than the best results for these models, clearly showing the value of the protolexicon and replicating the results found by Feldman et al. (2013a) on this dataset. Furthermore, TLD consistently outperforms the LD model, finding better phonetic categories, both for vowels generated from the combined categories of all speakers ('all') and vowels generated from adult female speakers only ('w'), although the latter are clearly much easier for both models to learn. Both models perform less well when the consonant frames provide less information, but the TLD model performance degrades less than the LD performance.

Both the TLD and the LD models find 'supervowel' categories, which cover multiple vowel categories and are used to merge minimal pairs into a single lexical item. Figure 4 shows example vowel categories inferred by the TLD model, including two supervowels. The TLD supervowels are used much less frequently than the supervowels found by the LD model, containing, on average, only two-thirds as many tokens.

Figure 5 shows that TLD also outperforms LD on the lexeme/word categorization task. Again performance decreases as the consonant categories become coarser, but the additional semantic information in the TLD model compensates for the lack of consonant information. In the individual components of VM, TLD and LD have similar VC ("recall"), but TLD has higher VH ("precision"), demonstrating that the semantic information given by the topics can separate potentially ambiguous words, as hypothesized.

Overall, the contextual semantic information

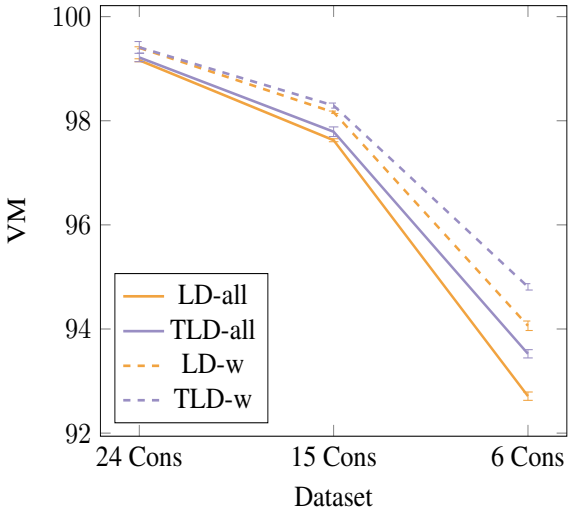


Figure 5: Lexeme evaluation. 'all' refers to datasets with vowels synthesized from all speakers, 'w' to datasets with vowels synthesized from adult female speakers' vowels.

added in the TLD model leads to both better phonetic categorization and to a better protolexicon, especially when the input is noisier, using degraded consonants. Since infants are not likely to have perfect knowledge of phonetic categories at this stage, semantic information is a potentially rich source of information that could be drawn upon to offset noise from other domains. The form of the semantic information added in the TLD model is itself quite weak, so the improvements shown here are in line with what infant learners could achieve.

7 Conclusion

Language acquisition is a complex task, in which many heterogeneous sources of information may be useful. In this paper, we investigated whether contextual semantic information could be of help when learning phonetic categories. We found that this contextual information can improve phonetic learning performance considerably, especially in situations where there is a high degree of phonetic ambiguity in the word-forms that learners hear. This suggests that previous models that have ignored semantic information may have underestimated the information that is available to infants. Our model illustrates one way in which language learners might harness the rich information that is present in the world without first needing to acquire a full inventory of word meanings.

The contextual semantic information that the TLD model tracks is similar to that potentially used in other linguistic learning tasks. Theories of cross-situational word learning (Smith and Yu, 2008; Yu and Smith, 2007) assume that sensitivity to situational co-occurrences between words and non-linguistic contexts is a precursor to learning the meanings of individual words. Under this view, contextual semantics is available to infants well before they have acquired large numbers of semantic minimal pairs. However, recent experimental evidence indicates that learners do not always retain detailed information about the referents that are present in a scene when they hear a word (Medina et al., 2011; Trueswell et al., 2013). This evidence poses a direct challenge to theories of cross-situational word learning. Our account does not necessarily require learners to track co-occurrences between words and individual objects, but instead focuses on more abstract information about salient events and topics in the environment; it will be important to investigate to what extent infants encode this information and use it in phonetic learning.

Regardless of the specific way in which infants encode semantic information, our method of adding this information by using LDA topics from transcript data was shown to be effective. This method is practical because it can approximate semantic information without relying on extensive manual annotation.

The LD model extended the phonetic categorization task by adding word contexts; the TLD model presented here goes even further, adding larger situational contexts. Both forms of top-down information help the low-level task of classifying acoustic signals into phonetic categories, furthering a holistic view of language learning with interaction across multiple levels.

Acknowledgments

This work was supported by EPSRC grant EP/H050442/1 and a James S. McDonnell Foundation Scholar Award to the final author.

References

Frans Adriaans and Daniel Swingley. Distributional learning of vowel categories is supported by prosody in infant-directed speech. In *Proceedings of the 34th Annual Conference of the Cognitive Science Society (CogSci)*, 2012.

E. Bergelson and D. Swingley. At 6-9 months, human infants know the meanings of many

common nouns. *Proceedings of the National Academy of Sciences*, 109(9):3253–3258, Feb 2012.

David M. Blei, Thomas L. Griffiths, Michael I. Jordan, and Joshua B. Tenenbaum. Hierarchical topic models and the nested Chinese restaurant process. In *Advances in Neural Information Processing Systems 16*, 2003.

Michael R. Brent and Jeffrey M. Siskind. The role of exposure to isolated words in early vocabulary development. *Cognition*, 81(2):B33–B44, 2001.

Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. Two decades of unsupervised POS induction: How far have we come? In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 575–584, Cambridge, MA, October 2010. Association for Computational Linguistics.

Bart de Boer and Patricia K. Kuhl. Investigating the role of infant-directed speech with a computer model. *Acoustics Research Letters Online*, 4(4): 129, 2003.

Brian Dillon, Ewan Dunbar, and William Idsardi. A single-stage approach to learning phonological categories: Insights from Inuktitut. *Cognitive Science*, 37(2):344–377, Mar 2013.

Micha Elsner, Sharon Goldwater, Naomi Feldman, and Frank Wood. A cognitive model of early lexical acquisition with phonetic variability. In *Proceedings of the 18th Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2013.

Naomi H. Feldman, Thomas L. Griffiths, Sharon Goldwater, and James L. Morgan. A role for the developing lexicon in phonetic category acquisition. *Psychological Review*, 2013a.

Naomi H. Feldman, Emily B. Myers, Katherine S. White, Thomas L. Griffiths, and James L. Morgan. Word-level information influences phonetic learning in adults and infants. *Cognition*, 127(3): 427–438, 2013b.

Abdellah Fourtassi and Emmanuel Dupoux. A rudimentary lexicon and semantics help bootstrap phoneme acquisition. Submitted.

Michael C. Frank, Noah D. Goodman, and Joshua B. Tenenbaum. Using speakers' referential intentions to model early cross-situational word learning. *Psychological Science*, 20(5): 578–585, 2009.

Manuela Friedrich and Angela D. Friederici. Word learning in 6-month-olds: Fast encoding—weak retention. *Journal of Cognitive Neuroscience*, 23 (11):3228–3240, Nov 2011.

Lakshmi J. Gogate and Lorraine E. Bahrick. Intersensory redundancy and 7-month-old infants' memory for arbitrary syllable-object relations. *Infancy*, 2(2):219–231, Apr 2001.

J. Hillenbrand, L. A. Getty, M. J. Clark, and K. Wheeler. Acoustic characteristics of American English vowels. *Journal of the Acoustical Society of America*, 97(5 Pt 1):3099–3111, May 1995.

P. W. Jusczyk and Elizabeth A. Hohne. Infants' memory for spoken words. *Science*, 277(5334): 1984–1986, Sep 1997.

Patricia K. Kuhl, Karen A. Williams, Francisco Lacerda, Kenneth N. Stevens, and Bjorn Lindblom. Linguistic experience alters phonetic perception in infants by 6 months of age. *Science*, 255(5044):606–608, 1992.

Brian MacWhinney. *The CHILDES Project: Tools for Analyzing Talk*. Lawrence Erlbaum Associates, 2000.

D. R. Mandel, P. W. Jusczyk, and D. B. Pisoni. Infants' recognition of the sound patterns of their own names. *Psychological Science*, 6(5):314–317, Sep 1995.

Nivedita Mani and Kim Plunkett. Twelve-month-olds know their cups from their keps and tups. *Infancy*, 15(5):445470, Sep 2010.

Jessica Maye, Daniel J. Weiss, and Richard N. Aslin. Statistical phonetic learning in infants: facilitation and feature generalization. *Developmental Science*, 11(1):122–134, Jan 2008.

Jessica Maye, Janet F Werker, and LouAnn Gerken. Infant sensitivity to distributional information can affect phonetic discrimination. *Cognition*, 82(3):B101–B111, Jan 2002.

Andrew McCallum. MALLET: A machine learning for language toolkit, 2002.

Bob McMurray, Richard N. Aslin, and Joseph C. Toscano. Statistical learning of phonetic categories: insights from a computational approach. *Developmental Science*, 12(3):369–378, May 2009.

Tamara Nicol Medina, Jesse Snedeker, John C. Trueswell, and Lila R. Gleitman. How words can and cannot be learned by observation. *Proceedings of the National Academy of Sciences*, 108(22):9014–9019, 2011.

Radford Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9: 249–265, 2000.

Linda Polka and Janet F. Werker. Developmental changes in perception of nonnative vowel contrasts. *Journal of Experimental Psychology: Human Perception and Performance*, 20(2):421–435, 1994.

Carl Rasmussen. The infinite Gaussian mixture model. In *Advances in Neural Information Processing Systems 13*, 2000.

Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 12th Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2007.

Brandon C. Roy, Michael C. Frank, and Deb Roy. Relating activity contexts to early word learning in dense longitudinal data. In *Proceedings of the 34th Annual Conference of the Cognitive Science Society (CogSci)*, 2012.

Rushen Shi and Janet F. Werker. The basis of preference for lexical words in 6-month-old infants. *Developmental Science*, 6(5):484–488, 2003.

M. Shukla, K. S. White, and R. N. Aslin. Prosody guides the rapid mapping of auditory word forms onto visual objects in 6-mo-old infants. *Proceedings of the National Academy of Sciences*, 108 (15):6038–6043, Apr 2011.

Linda B. Smith and Chen Yu. Infants rapidly learn word-referent mappings via cross-situational statistics. *Cognition*, 106(3):1558–1568, 2008.

Christine L. Stager and Janet F. Werker. Infants listen for more phonetic detail in speech perception than in word-learning tasks. *Nature*, 388: 381–382, 1997.

D. Swingley. Contributions of infant word learning to language development. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1536):3617–3632, Nov 2009.

Yee Whye Teh. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 985 – 992, Sydney, 2006.

Tuomas Teinonen, Richard N. Aslin, Paavo Alku, and Gergely Csibra. Visual speech contributes to phonetic learning in 6-month-old infants. *Cognition*, 108:850–855, 2008.

Erik D. Thiessen. The effect of distributional information on children's use of phonemic contrasts. *Journal of Memory and Language*, 56(1):16–34, Jan 2007.

R. Tincoff and P. W. Jusczyk. Some beginnings of word comprehension in 6-month-olds. *Psychological Science*, 10(2):172–175, Mar 1999.

Ruth Tincoff and Peter W. Jusczyk. Six-month-olds comprehend words that refer to parts of the body. *Infancy*, 17(4):432444, Jul 2012.

N. S. Trubetzkoy. *Grundzüge der Phonologie*. Vandenhoeck und Ruprecht, Göttingen, 1939.

John C. Trueswell, Tamara Nicol Medina, Alon Hafri, and Lila R. Gleitman. Propose but verify: Fast mapping meets cross-situational word learning. *Cognitive Psychology*, 66:126–156, 2013.

G. K. Vallabha, J. L. McClelland, F. Pons, J. F. Werker, and S. Amano. Unsupervised learning of vowel categories from infant-directed speech. *Proceedings of the National Academy of Sciences*, 104(33):13273–13278, Aug 2007.

Janet F. Werker and Richard C. Tees. Cross-language speech perception: Evidence for perceptual reorganization during the first year of life. *Infant Behavior and Development*, 7:49–63, 1984.

H. Henny Yeung and Janet F. Werker. Learning words' sounds before learning how words sound: 9-month-olds use distinct objects as cues to categorize speech information. *Cognition*, 113(2): 234–243, Nov 2009.

Chen Yu and Linda B. Smith. Rapid word learning under uncertainty via cross-situational statistics. *Psychological Science*, 18(5):414–420, 2007.

Bootstrapping into Filler-Gap: An Acquisition Story

Marten van Schijndel **Micha Elsner**
The Ohio State University
{vanschm,melsner}@ling.ohio-state.edu

Abstract

Analyses of filler-gap dependencies usually involve complex syntactic rules or heuristics; however recent results suggest that filler-gap comprehension begins earlier than seemingly simpler constructions such as ditransitives or passives. Therefore, this work models filler-gap acquisition as a byproduct of learning word orderings (e.g. SVO vs OSV), which must be done at a very young age anyway in order to extract meaning from language. Specifically, this model, trained on part-of-speech tags, represents the preferred locations of semantic roles relative to a verb as Gaussian mixtures over real numbers.

This approach learns role assignment in filler-gap constructions in a manner consistent with current developmental findings and is extremely robust to initialization variance. Additionally, this model is shown to be able to account for a characteristic error made by learners during this period (*A and B gorped* interpreted as *A gorped B*).

1 Introduction

The phenomenon of filler-gap, where the argument of a predicate appears outside its canonical position in the phrase structure (e.g. *[the apple]$_i$ that the boy ate t_i* or *[what]$_i$ did the boy eat t_i*), has long been an object of study for syntacticians (Ross, 1967) due to its apparent processing complexity. Such complexity is due, in part, to the arbitrary length of the dependency between a filler and its gap (e.g. *[the apple]$_i$ that Mary said the boy ate t_i*).

Recent studies indicate that comprehension of filler-gap constructions begins around 15 months (Seidl et al., 2003; Gagliardi et al., 2014). This finding raises the question of how such a complex phenomenon could be acquired so early since children at that age do not yet have a very advanced grasp of language (e.g. ditransitives do not seem to be generalized until at least 31 months; Goldberg et al. 2004, Bello 2012). This work shows that filler-gap comprehension in English may be acquired through learning word orderings rather than relying on hierarchical syntactic knowledge.

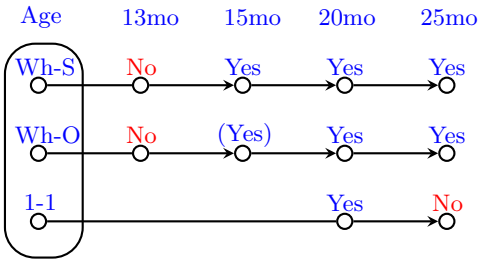


Figure 1: The developmental timeline of subject (Wh-S) and object (Wh-O) *wh*-clause extraction comprehension suggested by experimental results (Seidl et al., 2003; Gagliardi et al., 2014). Parentheses indicate weak comprehension. The final row shows the timeline of 1-1 role bias errors (Naigles, 1990; Gertner and Fisher, 2012). Missing nodes denote a lack of studies.

This work describes a cognitive model of the developmental timecourse of filler-gap comprehension with the goal of setting a lower bound on the modeling assumptions necessary for an ideal learner to display filler-gap comprehension. In particular, the model described in this paper takes chunked child-directed speech as input and learns orderings over semantic roles. These orderings then permit the model to successfully resolve filler-gap dependencies.[1] Further, the model presented here is also shown to initially reflect an idiosyncratic role assignment error observed in development (e.g. *A and B kradded* interpreted as *A kradded B*; Gertner and Fisher, 2012), though after training, the model is able to avoid the error. As such, this work may be said to model a learner from 15 months to between 25 and 30 months.

[1]This model does not explicitly learn gap positions, but rather assigns thematic roles to arguments based on where those arguments are expected to manifest. This approach to filler-gap comprehension is supported by findings that show people do not actually link fillers to gap positions but instead link the filler to a verb with missing arguments (Pickering and Barry, 1991)

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1084–1093,
Baltimore, Maryland, USA, June 23-25 2014.

2 Background

The developmental timeline during which children acquire the ability to process filler-gap constructions is not well-understood. Language comprehension precedes production, and the developmental literature on the acquisition of filler-gap constructions is sparsely populated due to difficulties in designing experiments to test filler-gap comprehension in preverbal infants. Older studies typically looked at verbal children and the mistakes they make to gain insight into the acquisition process (de Villiers and Roeper, 1995).

Recent studies, however, indicate that filler-gap comprehension likely begins earlier than production (Seidl et al., 2003; Gagliardi and Lidz, 2010; Gagliardi et al., 2014). Therefore, studies of verbal children are probably actually testing the acquisition of production mechanisms (planning, motor skills, greater facility with lexical access, etc) rather than the acquisition of filler-gap. Note that these may be related since filler-gap could introduce greater processing load which could overwhelm the child's fragile production capacity (Phillips, 2010).

Seidl et al. (2003) showed that children are able to process *wh*-extractions from subject position (e.g. *[who]$_i$ t_i ate pie*) as young as 15 months while similar extractions from object position (e.g. *[what]$_i$ did the boy eat t_i*) remain unparseable until around 20 months of age.[2] This line of investigation has been reopened and expanded by Gagliardi et al. (2014) whose results suggest that the experimental methodology employed by Seidl et al. (2003) was flawed in that it presumed infants have ideal performance mechanisms. By providing more trials of each condition and controlling for the pragmatic felicity of test statements, Gagliardi et al. (2014) provide evidence that 15-month old infants can process *wh*-extractions from both subject and object positions. Object extractions are more difficult to comprehend than subject extractions, however, perhaps due to additional processing load in object extractions (Gibson, 1998; Phillips, 2010). Similarly, Gagliardi and Lidz (2010) show that relativized extractions with a *wh*-relativizer (e.g. *find [the boy]$_i$ who t_i ate the apple*) are easier to comprehend than relativized extractions with *that* as the relativizer (e.g. *find [the boy]$_i$ that t_i ate the apple*).

Yuan et al. (2012) demonstrate that 19-month olds use their knowledge of nouns to learn both verbs and their associated argument structure. In their study, infants were shown video of a person talking on a phone using a nonce verb with either one or two nouns (e.g. *Mary kradded Susan*). Under the assumption that infants look longer at things that correspond to their understanding of a prompt, the infants were then shown two images that potentially depicted the described action – one picture where two actors acted independently (reflecting an intransitive proposition) and one picture where one actor acted on the other (reflecting a transitive proposition).[3] Even though the infants had no extralinguistic knowledge about the verb, they consistently treated the verb as transitive if two nouns were present and intransitive if only one noun was present.

Similarly, Gertner and Fisher (2012) show that intransitive phrases with conjoined subjects (e.g. *John and Mary gorped*) are given a transitive interpretation (i.e. *John gorped Mary*) at 21 months (henceforth termed '1-1 role bias'), though this effect is no longer present at 25 months (Naigles, 1990). This finding suggests both that learners will ignore canonical structure in favor of using all possible arguments and that children have a bias to assign a unique semantic role to each argument. It is important to note, however, that cross-linguistically children do not seem to generalize beyond two arguments until after at least 31 months of age (Goldberg et al., 2004; Bello, 2012), so a predicate occurring with three nouns would still likely be interpreted as merely transitive rather than ditransitive.

Computational modeling provides a way to test the computational level of processing (Marr, 1982). That is, given the input (child-directed speech, adult-directed speech, and environmental experiences), it is possible to probe the computational processes that result in the observed output. However, previous computational models of grammar induction (Klein and Manning, 2004), including infant grammar induction (Kwiatkowski et al., 2012), have not addressed filler-gap comprehension.[4]

The closest work to that presented here is the work on BabySRL (Connor et al., 2008; Connor et al., 2009; Connor et al., 2010). BabySRL is a computational model of semantic role acquistion using a similar set of assumptions to the current work. BabySRL learns weights over ordering constraints (e.g. preverbal, second noun, etc.) to acquire semantic role labelling while still exhibiting 1-1 role bias. However, no analysis has evaluated the abil-

[2] Since the *wh*-phrase is in the same (or a very similar) position as the original subject when the *wh*-phrase takes subject position, it is not clear that these constructions are true extractions (Culicover, 2013), however, this paper will continue to refer to them as such for ease of exposition.

[3] There were two actors in each image to avoid biasing the infants to look at the image with more actors.

[4] As one reviewer notes, Joshi et al. (1990) and subsequent work show that filler-gap phenomena can be formally captured by mildly context-sensitive grammar formalisms; these have the virtue of scaling up to adult grammar, but due to their complexity, do not seem to have been described as models of early acquisition.

Susan	said	**John**	gave	**girl**	**book**
-3	-2	-1	0	1	2

Table 1: An example of a chunked sentence (*Susan said John gave the girl a red book*) with the sentence positions labelled. Nominal heads of noun chunks are in bold.

ity of BabySRL to acquire filler-gap constructions. Further comparison to BabySRL may be found in Section 6.

3 Assumptions

The present work restricts itself to acquiring filler-gap comprehension in English. The model presented here learns a single, non-recursive ordering for the semantic roles in each sentence relative to the verb since several studies have suggested that early child grammars may consist of simple linear grammars that are dictated by semantic roles (Diessel and Tomasello, 2001; Jackendoff and Wittenberg, in press). This work assumes learners can already identify nouns and verbs, which is supported by Shi et al. (1999) who show that children at an extremely young age can distinguish between content and function words and by Waxman and Booth (2001) who show that children can distinguish between different types of content words. Further, since Waxman and Booth (2001) demonstrate that, by 14 months, children are able to distinguish nouns from modifiers, this work assumes learners can already chunk nouns and access the nominal head. To handle recursion, this work assumes that children treat the final verb in each sentence as the main verb (implicitly assuming sentence segmentation), which ideally assigns roles to each of the nouns in the sentence.

Due to the findings of Yuan et al. (2012), this work adopts a 'syntactic bootstrapping' theory of acquisition (Gleitman, 1990), where structural properties (e.g. number of nouns) inform the learner about semantic properties of a predicate (e.g. how many semantic roles it confers). Since infants infer the number of semantic roles, this work further assumes they already have expectations about where these roles tend to be realized in sentences, if they appear. These positions may correspond to different semantic roles for different predicates (e.g. the subject of *run* and of *melt*); however, the role for predicates with a single argument is usually assigned to the noun that precedes the verb while a second argument is usually assigned after the verb. The semantic properties of these roles may be learned lexically for each predicate, but that is beyond the scope of this work. Therefore, this work uses syntactic and semantic roles interchangeably (e.g. *subject* and *agent*).

	μ	σ	π
G_{SC}	-1	0.5	.999
G_{SN}	-1	3	.001
G_{OC}	1	0.5	.999
G_{ON}	1	3	.001
Φ	.00001		

Table 2: Initial values for the mean (μ), standard deviation (σ), and prior (π) of each Gaussian as well as the skip penalty (Φ) used in this paper.

Finally, following the finding by Gertner and Fisher (2012) that children interpret intransitives with conjoined subjects as transitives, this work assumes that semantic roles have a one-to-one correspondence with nouns in a sentence (similarly used as a soft constraint in the semantic role labelling work of Titov and Klementiev, 2012).

4 Model

The model represents the preferred locations of semantic roles relative to the verb as distributions over real numbers. This idea is adapted from Boersma (1997) who uses it to learn constraint rankings in optimality theory.

In this work, the final (main) verb is placed at position 0; words (and chunks) before the verb are given progressively more negative positions, and words after the verb are given progressively more positive positions (see Table 1). Learner expectations of where an argument will appear relative to the verb are modelled as two-component Gaussian mixtures: one mixture of Gaussians ($G_{S\cdot}$) corresponds to the subject argument, another ($G_{O\cdot}$) corresponds to the object argument. There is no mixture for a third argument since children do not generalize beyond two arguments until later in development (Goldberg et al., 2004; Bello, 2012).

One component of each mixture learns to represent the canonical position for the argument ($G_{\cdot C}$) while the other ($G_{\cdot N}$) represents some alternate, non-canonical position such as the filler position in filler-gap constructions. To reflect the fact that learners have had 15 months of exposure to their language before acquiring filler-gap, the mixture is initialized so that there is a stronger probability associated with the canonical Gaussian than with the non-canonical Gaussian of each mixture.[5] Finally, the one-to-one role bias is explicitly encoded such that the model cannot use a label that has already been used elsewhere in the sentence.

[5] Akhtar (1999) finds that learners may not have strong expectations of canonical argument positions until four years of age, but the results of the current study are extremely robust to changes in initialization, as discussed in Section 7 of this paper, so this assumption is mostly adopted for ease of exposition.

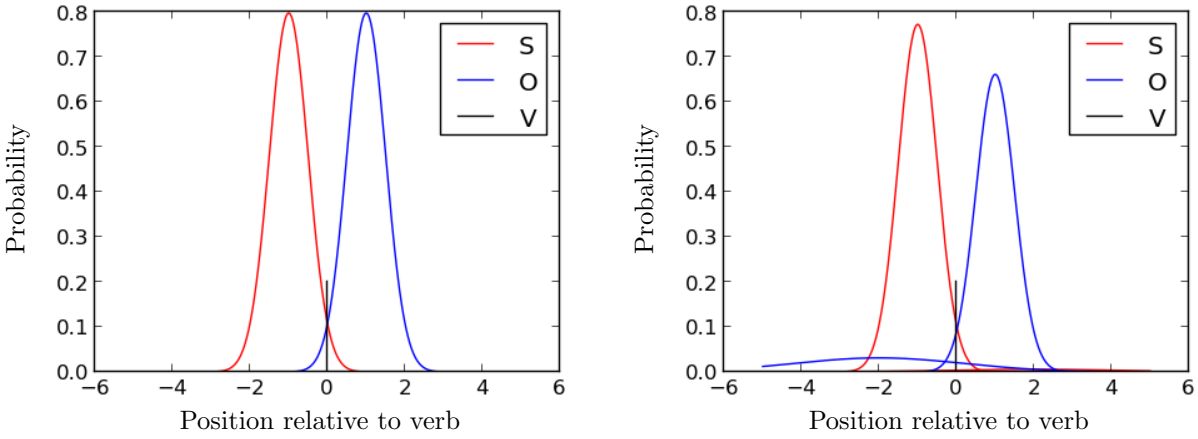


Figure 2: Visual representations of (Left) the initial model's expectations of where arguments will appear, given the initial parameters in Table 2 and (Right) the converged model's expectations of where arguments will appear.

Thus, the initial model conditions (see Figure 2) are most likely to realize an SVO ordering, although it is possible to obtain SOV (by sampling a negative number from the blue curve) or even OSV (by also sampling the red curve very close to 0). The model is most likely to hypothesize a preverbal object when it has already assigned the subject role to something and, in addition, there is no postverbal noun competing for the object label. In other words, the model infers that an object extraction may have occurred if there is a 'missing' postverbal argument.

Finally, the probability of a given sequence is the product of the label probabilities for the component argument positions (e.g. G_{SC} generating an argument at position -2, etc). Since many sentences have more than two nouns, the model is allowed to skip nouns by multiplying a penalty term (Φ) into the product for each skipped noun; the cost is set at 0.00001 for this study, though see Section 7 for a discussion of the constraints on this parameter. See Table 2 for initialization parameters and Figure 2 for a visual representation of the initial expectations of the model.

This work uses a model with 2-component mixtures for both subjects and objects (termed the *symmetric model*). This formulation achieves the best fit to the training data according to the Bayesian Information Criterion (BIC).[6] However, follow-up experiments find that the non-canonical subject Gaussian only improves the likelihood of the data by erroneously modeling postverbal nouns in imperative statements. The lack of a canonical subject in English imperatives allows the model to improve the likelihood of the data by using the non-canonical subject Gaussian to capture fictitious postverbal arguments. When imperatives are filtered out of the training corpus, the symmetric model obtains a worse BIC fit than a model that lacks the non-canonical subject Gaussian. Therefore, if one makes the assumption that imperatives are prosodically-marked for learners (e.g. the learner is the implicit subject), the best model is one that lacks a non-canonical subject.[7] The remainder of this paper assumes a symmetric model to demonstrate what happens if such an assumption is not made; for the evaluations described in this paper, the results are similar in either case.

This model differs from other non-recursive computational models of grammar induction (e.g. Goldwater and Griffiths, 2007) since it is not based on Hidden Markov Models. Instead, it determines the best ordering for the sentence as a whole. This approach bears some similarity to a Generalized Mallows model (Chen et al., 2009), but the current formulation was chosen due to being independently posited as cognitively plausible (Boersma, 1997).

Figure 2 (Right) shows the converged, final state of the model. The model expects the first argument (usually agent) to be assigned preverbally and expects the second (say, patient) to be assigned postverbally; however, there is now a larger chance that the second argument will appear preverbally.

5 Evaluation

The model in this work is trained using transcribed child-directed speech (CDS) from the BabySRL portions (Connor et al., 2008) of CHILDES (MacWhinney, 2000). Chunking is performed us-

[6]The BIC rewards improved log-likelihood but penalizes increased model complexity.

[7]This finding suggests that a Dirichlet Process or other means of dynamically determining the number of components in each mixture would converge to a model that lacks non-canonical subjects if imperative filtering were employed.

	Eve (n = 4820)			Adam (n = 4461)		
	P	R	F	P	R	F
Initial	.54	.64	.59	.53	.60	.56
Trained	.52	.69	.59*	.51	.65	.57*
Initial$_c$	.56	.66	.60	.55	.62	.58
Trained$_c$	.54	.71	.61*	.53	.67	.59*

Table 3: Overall accuracy on the Eve and Adam sections of the BabySRL corpus. Bottom rows reflect accuracy when non-agent roles are collapsed into a single role. Note that improvements are numerically slight since filler-gap is relatively rare (Schuler, 2011). $^*p << .01$

Subject Extraction filter:	S	x	V	...
Object Extraction filter:	O	...	V	...

	Eve (n = 1345)			Adam (n = 1287)		
	P	R	F	P	R	F
Initial$_c$	.53	.57	.55	.53	.52	.52
Trained$_c$	.55	.67	.61*	.54	.63	.58*

Table 4: (Above) Filters to extract filler-gap constructions: A) the subject and verb are not adjacent, B) the object precedes the verb. (Below) Filler-gap accuracy on the Eve and Adam sections of the BabySRL corpus when non-agent roles are collapsed into a single role. $^*p << .01$

ing a basic noun-chunker from NLTK (Bird et al., 2009). Based on an initial analysis of chunker performance, *yes* is hand-corrected to not be a noun. Poor chunker perfomance is likely due to a mismatch in chunker training and testing domains (Wall Street Journal text vs transcribed speech), but chunking noise may be a good estimation of learner uncertainty, so the remaining text is left uncorrected. All noun phrase chunks are then replaced with their final noun (presumed the head) to approximate the ability of children to distinguish nouns from modifiers (Waxman and Booth, 2001). Finally, for each sentence, the model assigns sentence positions to each word with the final verb at zero.

Viterbi Expectation-Maximization is performed over each sentence in the corpus to infer the parameters of the model. During the Expectation step, the model uses the current Gaussian parameters to label the nouns in each sentence with argument roles. Since the model is not lexicalized, these roles correspond to the semantic roles most commonly associated with subject and object. The model then chooses the best label sequence for each sentence.

These newly labelled sentences are used during the Maximization step to determine the Gaussian parameters that maximize the likelihood of that labelling. The mean of each Gaussian is updated to the mean position of the words it labels. Similarly, the standard deviation of each Gaussian is updated with the standard deviation of the positions it labels. A learning rate of 0.3 is used to prevent large parameter jumps. The prior probability of each Gaussian is updated as the ratio of that Gaussian's labellings to the total number of labellings from that mixture in the corpus:

$$\pi_{\rho\theta} = \frac{|\ G_{\rho\theta}\ |}{|\ G_{\rho\cdot}\ |} \quad (1)$$

where $\rho \in \{S, O\}$ and $\theta \in \{C, N\}$.

Best results seem to be obtained when the skip-penalty is loosened by an order of magnitude during testing. Essentially, this forces the model to tightly adhere to the perceived argument structure during training to learn more rigid parameters, but the model is allowed more leeway to skip arguments it has less confidence in during testing. Convergence (see Figure 2) tends to occur after four iterations but can take up to ten iterations depending on the initial parameters.

Since the model is unsupervised, it is trained on a given corpus (e.g. Eve) before being tested on the role annotations of that same corpus. The Eve corpus was used for development purposes,[8] and the Adam data was used only for testing.

For testing, this study uses the semantic role annotations in the BabySRL corpus. These annotations were obtained by automatically semantic role labelling portions of CHILDES with the system of Punyakanok et al. (2008) before roughly hand-correcting them (Connor et al., 2008). The BabySRL corpus is annotated with 5 different roles, but the model described in this paper only uses 2 roles. Therefore, overall accuracy results (see Table 3) are presented both for the raw BabySRL corpus and for a collapsed BabySRL corpus where all non-agent roles are collapsed into a single role (denoted by a subscript $_c$ in all tables).

Since children do not generalize above two arguments during the modelled age range (Goldberg et al., 2004; Bello, 2012), the collapsed numbers more closely reflect the performance of a learner at this age than the raw numbers. The increase in accuracy obtained from collapsing non-agent arguments indicates that children may initially generalize incorrectly to some verbs and would need to learn lexically-specific role assignments (e.g. double-object constructions of *give*). Since the current work is interested in general filler-gap comprehension at this age, including over unknown verbs, the remaining analyses in this paper con-

[8]This is included for transparency, though the initial parameters have very little bearing on the final results as stated in Section 7, so the danger of overfitting to development data is very slight.

	P	R	F	P	R	F
Eve	Subj (n = 691)			Obj (n = 654)		
Initial_c	.66	.83	.74	.35	.31	.33
Trained_c	.64	.84	$.72^{\dagger}$	.45	.52	$.48^{*}$
Adam	Subj (n = 886)			Obj (n = 1050)		
Initial_c	.69	.81	.74	.33	.27	.30
Trained_c	.66	.81	.73	.44	.48	$.46^{*}$

	P	R	F	P	R	F
Eve	Wh- (n = 689)			That (n = 125)		
Initial_c	.63	.45	.53	.43	.48	.45
Trained_c	.73	.75	$.74^{*}$	.44	.57	$.50^{\dagger}$
Adam	Wh- (n = 748)			That (n = 189)		
Initial_c	.50	.37	.42	.50	.50	.50
Trained_c	.61	.65	$.63^{*}$	.47	.56	$.51^{\dagger}$

Table 5: (Left) Subject-extraction accuracy and object-extraction accuracy and (Right) *Wh*-relative accuracy and *that*-relative accuracy; calculated over the Eve and Adam sections of the BabySRL corpus with non-agent roles collapsed into a single role. $^{\dagger}p = .02$ $^{*}p << .01$

sider performance when non-agent arguments are collapsed.[9]

Next, a filler-gap version of the BabySRL corpus is created using a coarse filtering process: the new corpus is comprised of all sentences where an associated object precedes the final verb and all sentences where the relevant subject is not immediately followed by the final verb (see Table 4). For these filler-gap evaluations, the model is trained on the full version of the corpus in question (e.g. Eve) before being tested on the filler-gap subset of that corpus. The overall results of the filler-gap evaluation (see Table 4) indicate that the model improves significantly at parsing filler-gap constructions after training.

The performance of the model on role-assignment in filler-gap constructions may be analyzed further in terms of how the model performs on subject-extractions compared with object-extractions and in terms of how the model performs on *that*-relatives compared with *wh*-relatives (see Table 5).

The model actually performs worse at subject-extractions after training than before training. This is unsurprising because, prior to training, subjects have little-to-no competition for preverbal role assignments; after training, there is a preverbal extracted object category, which the model can erroneously use. This slight, though significant in Eve, deficit is counter-balanced by a very substantial and significant improvement in object-extraction labelling accuracy.

Similarly, training confers a large and significant improvement for role assignment in *wh*-relative constructions, but it yields less of an improvement for *that*-relative constructions. This difference mimics a finding observed in the developmental literature where children seem slower to acquire comprehension of *that*-relatives than of *wh*-relatives (Gagliardi and Lidz, 2010).

[9]Though performance is slightly worse when arguments are not collapsed, all the same patterns emerge.

6 Comparison to BabySRL

The acquisition of semantic role labelling (SRL) by the BabySRL model (Connor et al., 2008; Connor et al., 2009; Connor et al., 2010) bears many similarities to the current work and is, to our knowledge, the only comparable line of inquiry to the current one. The primary function of BabySRL is to model the acquisition of semantic role labelling while making an idiosyncratic error which infants also make (Gertner and Fisher, 2012), the 1-1 role bias error (*John and Mary gorped* interpreted as *John gorped Mary*). Similar to the model presented in this paper, BabySRL is based on simple ordering features such as argument position relative to the verb and argument position relative to the other arguments.

This section will demonstrate that the model in this paper initially reflects 1-1 role bias comparably to BabySRL, though it progresses beyond this bias after training.[10] Further, the model in this paper is able to reflect the concurrent acquisition of filler-gap whereas BabySRL does not seem well-suited to such a task. Finally, BabySRL performs undesirably in intransitive settings whereas the model in this paper does not.

Connor et al. (2008) demonstrate that a supervised perceptron classifier, based on positional features and trained on the silver role label annotations of the BabySRL corpus, manifests 1-1 role bias errors. Follow-up studies show that supervision may be lessened (Connor et al., 2009) or removed (Connor et al., 2010) and BabySRL will still reflect a substantial 1-1 role bias.

Connor et al. (2008) and Connor et al. (2009) run direct analyses of how frequently their models make 1-1 role bias errors. A comparable evaluation may be run on the current model by generating 1000 sentences with a structure of NNV and reporting how many times the model chooses a subject-first labelling (see Table 6).[11]

[10]All evaluations in this section are preceded by training on the chunked Eve corpus.

[11]While Table 6 analyzes erroneous labellings of NNV structure, the 'Obj' column of Table 5 (Left)

	Error rate
Initial	.36
Trained	.11
Initial (given 2 args)	.66
Trained (given 2 args)	.13
2008 arg-arg position	.65
2008 arg-verb position	0
2009 arg-arg position	.82
2009 arg-verb position	.63

Table 6: 1-1 role bias error in this model compared to the models of Connor et al. (2008) and Connor et al. (2009). That is, how frequently each model labelled an NNV sentence SOV. Since the Connor et al. models are perceptron-based, they require both arguments be labelled. The model presented in this paper does not share this restriction, so the raw error rate for this model is presented in the first two lines; the error rate once this additional restriction is imposed is given in the second two lines.

The results of Connor et al. (2008) and Connor et al. (2009) depend on whether BabySRL uses argument-argument relative position as a feature or argument-verb relative position as a feature (there is no combined model). Further, the model presented here from Connor et al. (2009) has a unique argument constraint, similar to the model in this paper, in order to make comparison as direct as possible.

The 1-1 role bias error rate (before training) of the model presented in this paper is comparable to that of Connor et al. (2008) and Connor et al. (2009), which shows that the current model provides comparable developmental modeling benefits to the BabySRL models. Further, similar to real children (see Figure 1) the model presented in this paper develops beyond this error by the end of its training,[12] whereas the BabySRL models still make this error after training.

Connor et al. (2010) look at how frequently their model correctly labels the agent in transitive and intransitive sentences with unknown verbs (to demonstrate that it exhibits an agent-first bias). This evaluation can be replicated for the current study by generating 1,000 sentences with the transitive form of NVN and a further 1,000 sentences with the intransitive form of NV (see Table 7).

Since Connor et al. (2010) investigate the effects of different initial lexicons, this evaluation compares against the resulting BabySRL from each initializer: they initially seed their part-of-speech tagger with either the 10 or 365 most frequent nouns in the corpus or they dispense with the tagger and use gold part-of-speech tags.

	NVN	NV
Sents in Eve	1173	1513
Sents in Adam	1029	1353
Initial	.67	1
Trained	.65	.96
Weak (10) lexical	.71	.59
Strong (365) lexical	.74	.41
Gold Args	.77	.58

Table 7: Agent-prediction recall accuracy in transitive (NVN) and intransitive (NV) settings of the model presented in this paper (middle) and the combined model of Connor et al. (2010) (bottom), which has features for argument-argument relative position as well as argument-predicate relative position and so is closest to the model presented in this paper.

As with subject extraction, the model in this paper gets less accurate after training because of the newly minted extracted object category that can be mistakenly used in these canonical settings. While the model of Connor et al. (2010) outperforms the model presented here when in a transitive setting, their model does much worse in an intransitive setting. The difference in transitive settings stems from increased lexicalization, as is apparent from their results alone; the model presented here initially performs close to their weakly lexicalized model, though training impedes agent-prediction accuracy due to an increased probability of non-canonical objects.

For the intransitive case, however, whereas the model presented in this paper is generally able to successfully label the lone noun as the subject, the model of Connor et al. (2010) chooses to label lone nouns as objects about 40% of the time. This likely stems from their model's reliance on argument-argument relative position as a feature; when there is no additional argument to use for reference, the model's accuracy decreases. This is borne out by their model (not shown in Table 7) that omits the argument-argument relative position feature and solely relies on verb-argument position, which achieves up to 70% accuracy in intransitive settings. Even in that case, however, BabySRL still chooses to label lone nouns as objects 30% of the time. The fact that intransitive sentences are more common than transitive sentences in both the Eve and Adam sections of the BabySRL corpus suggests that learners should be more likely to assign

shows model accuracy on NNV structures.

[12] It is important to note that the unique argument constraint prevents the current model from actually getting the correct, conjoined-subject parse, but it no longer exhibits agent-first bias, an important step for acquiring passives, which occurs between 3 and 4 years (Thatcher et al., 2008).

correct roles in an intransitive setting, which is not reflected in the BabySRL results.

The overall reason for the different results between the current work and BabySRL is that BabySRL relies on positional features that measure the relative position of two individual elements (e.g. where a given noun is relative to the verb). Since the model in this paper operates over global orderings, it implicitly takes into account the positions of other nouns as it models argument position relative to the verb; object and subject are in competition as labels for preverbal nouns, so a preverbal object is usually only assigned once a subject has already been detected.

Further, while BabySRL consistently reflects 1-1 role bias (corresponding to a pre 25-month old learner), it also learns to productively label five roles, which developmental studies have shown does not take place until at least 31 months (Goldberg et al., 2004; Bello, 2012). Finally, it does not seem likely that BabySRL could be easily extended to capture filler-gap acquisition. The argument-verb position features impede acquisition of filler-gap by classifying preverbal arguments as agents, and the argument-argument position features inhibit accurate labelling in intransitive settings and result in an agent-first bias which would tend to label extracted objects as agents. In fact, these observations suggest that any linear classifier which relies on positioning features will have difficulties modeling filler-gap acquisition.

In sum, the unlexicalized model presented in this paper is able to achieve greater labelling accuracy than the lexicalized BabySRL models in intransitive settings, though this model does perform slightly worse in the less common transitive setting. Further, the unsupervised model in this paper initially reflects developmental 1-1 role bias as well as the supervised BabySRL models, and it is able to progress beyond this bias. Finally, unlike BabySRL, the model presented here provides a cognitive model of the acquisition of filler-gap comprehension, which BabySRL does not seem well-suited to model.

7 Discussion

This paper has presented a simple cognitive model of filler-gap acquisition, which is able to capture several findings from developmental psychology. Training significantly improves role labelling in the case of object-extractions, which improves the overall accuracy of the model. This boost is accompanied by a slight decrease in labelling accuracy in subject-extraction settings. The asymmetric ease of subject versus object comprehension is well-documented in both children and adults (Gibson, 1998), and while training improves the model's ability to process object-extractions, there is still a gap between object-extraction and subject-extraction comprehension even after training.

Further, the model exhibits better comprehension of *wh*-relatives than *that*-relatives similar to children (Gagliardi and Lidz, 2010). This could also be an area where a lexicalized model could do better. As Gagliardi and Lidz (2010) point out, whereas *wh*-relatives such as *who* or *which* always signify a filler-gap construction, *that* can occur for many different reasons (demonstrative, determiner, complementizer, etc) and so is a much weaker filler-gap cue. A lexical model could potentially pick up on clues which could indicate when *that* is a relativizer or simply improve on its comprehension of *wh*-relatives even more.

It is interesting to note that the cuurent model does not make use of *that* as a cue at all and yet is still slower at acquiring *that*-relatives than *wh*-relatives. This fact suggests that the findings of Gagliardi and Lidz (2010) may be partially explained by a frequency effect: perhaps the input to children is simply biased such that *wh*-relatives are much more common than *that*-relatives (as shown in Table 5).

This model also initially reflects the 1-1 role bias observed in children (Gertner and Fisher, 2012) as well as previous models (Connor et al., 2008; Connor et al., 2009; Connor et al., 2010) without sacrificing accuracy in canonical intransitive settings.

Finally, this model is extremely robust to different initializations. The canonical Gaussian expectations can begin far from the verb (± 3) or close to the verb (± 0.1), and the standard deviations of the distributions and the skip-penalty can vary widely; the model always converges to give comparable results to those presented here. The only constraint on the initial parameters is that the probability of the extracted object occurring preverbally must exceed the skip-penalty (i.e. extraction must be possible). In short, this paper describes a simple, robust cognitive model of the development of a learner between 15 months until somewhere between 25- and 30-months old (since 1-1 role bias is no longer present but no more than two arguments are being generalized).

In future, it would be interesting to incorporate lexicalization into the model presented in this paper, as this feature seems likely to bridge the gap between this model and BabySRL in transitive settings. Lexicalization should also help further distinguish modifiers from arguments and improve the overall accuracy of the model.

It would also be interesting to investigate how well this model generalizes to languages besides English. Since the model is able to use the verb position as a semi-permeable boundary between canonical subjects and objects, it may not work as

well in verb-final languages, and thus makes the prediction that filler-gap comprehension may be acquired later in development in such languages due to a greater reliance on hierarchical syntax.

Ordering is one of the definining characteristics of a language that must be acquired by learners (e.g. SVO vs SOV), and this work shows that filler-gap comprehension can be acquired as a by-product of learning orderings rather than having to resort to higher-order syntax. Note that this model cannot capture the constraints on filler-gap usage which require a hierarchical grammar (e.g. subjacency), but such knowledge is really only needed for successful production of filler-gap constructions, which occurs much later (around 5 years; de Villiers and Roeper, 1995). Further, the kind of ordering system proposed in this paper may form an initial basis for learning such grammars (Jackendoff and Wittenberg, in press).

8 Acknowledgements

Thanks to Peter Culicover, William Schuler, Laura Wagner, and the attendees of the OSU 2013 Fall Linguistics Colloquium Fest for feedback on this work. This work was partially funded by an OSU Dept. of Linguistics Targeted Investment for Excellence (TIE) grant for collaborative interdisciplinary projects conducted during the academic year 2012-13.

References

Nameera Akhtar. 1999. Acquiring basic word order: evidence for data-driven learning of syntactic structure. *Journal of Child Language*, 26:339–356.

Sophia Bello. 2012. Identifying indirect objects in French: An elicitation task. In *Proceedings of the 2012 annual conference of the Canadian Linguistic Association.*

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit.* O'Reilly, Beijing.

Paul Boersma. 1997. How we learn variation, optionality, and probability. *Proceedings of the Institute of Phonetic Sciences of the University of Amsterdam*, 21:43–58.

Harr Chen, S.R.K. Branavan, Regina Barzilay, and David R. Karger. 2009. Content modeling using latent permutations. *Journal of Artificial Intelligence Research*, 36:129–163.

Michael Connor, Yael Gertner, Cynthia Fisher, and Dan Roth. 2008. Baby srl: Modeling early language acquisition. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning.*

Michael Connor, Yael Gertner, Cynthia Fisher, and Dan Roth. 2009. Minimally supervised model of early language acquisition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning.*

Michael Connor, Yael Gertner, Cynthia Fisher, and Dan Roth. 2010. Starting from scratch in semantic role labelling. In *Proceedings of ACL 2010.*

Peter Culicover. 2013. *Explaining syntax: representations, structures, and computation.* Oxford University Press.

Jill de Villiers and Thomas Roeper. 1995. Barriers, binding, and acquisition of the dp-np distinction. *Language Acquisition*, 4(1):73–104.

Holger Diessel and Michael Tomasello. 2001. The acquisition of finite complement clauses in english: A corpus-based analysis. *Cognitive Linguistics*, 12:1–45.

Annie Gagliardi and Jeffrey Lidz. 2010. Morphosyntactic cues impact filler-gap dependency resolution in 20- and 30-month-olds. In *Poster session of BUCLD35.*

Annie Gagliardi, Tara M. Mease, and Jeffrey Lidz. 2014. Discontinuous development in the acquisition of filler-gap dependencies: Evidence from 15- and 20-month-olds. Harvard unpublished manuscript: http://www.people.fas.harvard.edu/∼gagliardi.

Yael Gertner and Cynthia Fisher. 2012. Predicted errors in children's early sentence comprehension. *Cognition*, 124:85–94.

Edward Gibson. 1998. Linguistic complexity: Locality of syntactic dependencies. *Cognition*, 68(1):1–76.

Lila R. Gleitman. 1990. The structural sources of verb meanings. *Language Acquisition*, 1:3–55.

Adele E. Goldberg, Devin Casenhiser, and Nitya Sethuraman. 2004. Learning argument structure generalizations. *Cognitive Linguistics*, 14(3):289–316.

Sharon Goldwater and Tom Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics.*

Ray Jackendoff and Eva Wittenberg. in press. What you can say without syntax: A hierarchy of grammatical complexity. In Fritz Newmeyer and Lauren Preston, editors, *Measuring Linguistic Complexity.* Oxford University Press.

Aravind K. Joshi, K. Vijay Shanker, and David Weir. 1990. The convergence of mildly context-sensitive grammar formalisms. Technical Report MS-CIS-90-01, Department of Computer and Information Science, University of Pennsylvania.

Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*.

Tom Kwiatkowski, Sharon Goldwater, Luke S. Zettlemoyer, and Mark Steedman. 2012. A probabilistic model of syntactic and semantic acquisition from child-directed utterances and their meanings. In *Proceedings of EACL 2012*.

Brian MacWhinney. 2000. *The CHILDES project: Tools for analyzing talk*. Lawrence Elrbaum Associates, Mahwah, NJ, third edition.

David Marr. 1982. *Vision. A Computational Investigation into the Human Representation and Processing of Visual Information*. W.H. Freeman and Company.

Letitia R. Naigles. 1990. Children use syntax to learn verb meanings. *The Journal Child Language*, 17:357–374.

Colin Phillips. 2010. Some arguments and non-arguments for reductionist accounts of syntactic phenomena. *Language and Cognitive Processes*, 28:156–187.

Martin Pickering and Guy Barry. 1991. Sentence processing without empty categories. *Language and Cognitive Processes*, 6(3):229–259.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

John R. Ross. 1967. *Constraints on Variables in Syntax*. Ph.D. thesis, Massachusetts Institute of Technology.

William Schuler. 2011. Effects of filler-gap dependencies on working memory requirements for parsing. In *Proceedings of COGSCI*, pages 501–506, Austin, TX. Cognitive Science Society.

Amanda Seidl, George Hollich, and Peter W. Jusczyk. 2003. Early understanding of subject and object wh-questions. *Infancy*, 4(3):423–436.

Rushen Shi, Janet F. Werker, and James L. Morgan. 1999. Newborn infants' sensitivity to perceptual cues to lexical and grammatical words. *Cognition*, 72(2):B11–B21.

Katherine Thatcher, Holly Branigan, Janet McLean, and Antonella Sorace. 2008. Children's early acquisition of the passive: Evidence from syntactic priming. In *Proceedings of the Child Language Seminar 2007*, pages 195–205, University of Reading.

Ivan Titov and Alexandre Klementiev. 2012. Crosslingual induction of semantic roles. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL-2011)*.

Sandra R. Waxman and Amy E. Booth. 2001. Seeing pink elephants: Fourteen-month-olds' interpretations of novel nouns and adjectives. *Cognitive Psychology*, 43:217–242.

Sylvia Yuan, Cynthia Fisher, and Jesse Snedeker. 2012. Counting the nouns: Simple structural cues to verb meaning. *Child Development*, 83(4):1382–1399.

Nonparametric Learning of Phonological Constraints in Optimality Theory

Gabriel Doyle
Department of Linguistics
UC San Diego
La Jolla, CA, USA 92093
gdoyle@ucsd.edu

Klinton Bicknell
Department of Linguistics
Northwestern University
Evanston, IL, USA 60208
kbicknell@northwestern.edu

Roger Levy
Department of Linguistics
UC San Diego
La Jolla, CA, USA 92093
rlevy@ucsd.edu

Abstract

We present a method to jointly learn features and weights directly from distributional data in a log-linear framework. Specifically, we propose a non-parametric Bayesian model for learning phonological markedness constraints directly from the distribution of input-output mappings in an Optimality Theory (OT) setting. The model uses an Indian Buffet Process prior to learn the feature values used in the log-linear method, and is the first algorithm for learning phonological constraints without presupposing constraint structure. The model learns a system of constraints that explains observed data as well as the phonologically-grounded constraints of a standard analysis, with a violation structure corresponding to the standard constraints. These results suggest an alternative data-driven source for constraints instead of a fully innate constraint set.

1 Introduction

Many aspects of human cognition involve the interaction of constraints that push a decision-maker toward different options, whether in something so trivial as choosing a movie or so important as a fight-or-flight response. These constraint-driven decisions can be modeled with a log-linear system. In these models, a set of constraints is weighted and their violations are used to determine a probability distribution over outcomes. But where do these constraints come from?

We consider this question by examining the dominant framework in modern phonology, Optimality Theory (Prince and Smolensky, 1993, OT), implemented in a log-linear framework, MaxEnt OT (Goldwater and Johnson, 2003), with output forms' probabilities based on a weighted sum of constraint violations. OT analyses generally assume that the constraints are innate and universal, both to obviate the problem of learning constraints' identities and to limit the set of possible languages.

We propose a new approach: to learn constraints with limited innate phonological knowledge by identifying sets of constraint violations that explain the observed distributional data, instead of selecting constraints from an innate set of constraint definitions. Because the constraints are identified as sets of violations, this also permits constraints specific to a given language to be learned. This method, which we call IBPOT, uses an Indian Buffet Process (IBP) prior to define the space of possible constraint violation matrices, and uses Bayesian reasoning to identify constraint matrices likely to have generated the observed data. In identifying constraints solely by their extensional violation profiles, this method does not directly identify the intensional definitions of the identified constraints, but to the extent that the resulting violation profiles are phonologically interpretable, we may conclude that the data themselves guide constraint identification. We test IBPOT on tongue-root vowel harmony in Wolof, a West African language.

The set of constraints learned by the model satisfy two major goals: they explain the data as well as the standard phonological analysis, and their violation structures correspond to the standard constraints. This suggests an alternative data-driven genesis for constraints, rather than the traditional assumption of fully innate constraints.

2 Phonology and Optimality Theory

2.1 OT structure

Optimality Theory has been used for constraint-based analysis of many areas of language, but we focus on its most successful application: phonology. We consider an OT analysis of the mappings

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1094–1103,
Baltimore, Maryland, USA, June 23-25 2014.

between underlying forms and their phonological manifestations – i.e., mappings between forms in the mental lexicon and the actual vocalized forms of the words.[1]

Stated generally, an OT system takes some input, generates a set of candidate outputs, determines what constraints each output violates, and then selects a candidate output with a relatively unobjectionable violation profile. To do this, an OT system contains four major components: a generator GEN, which generates candidate output forms for the input; a set of constraints CON, which penalize candidates; a evaluation method EVAL, which selects an winning candidate; and H, a language-particular weighting of constraints that EVAL uses to determine the winning candidate. Previous OT work has focused on identifying the appropriate formulation of EVAL and the values and acquisition of H, while taking GEN and CON as given. Here, we expand the learning task by proposing an acquisition method for CON.

To learn CON, we propose a data-driven markedness constraint learning system that avoids both innateness and tractability issues. Unlike previous OT learning methods, which assume known constraint definitions and only learn the relative strength of these constraints, the IBPOT learns constraint violation profiles and weights for them simultaneously. The constraints are derived from sets of violations that effectively explain the observed data, rather than being selected from a preexisting set of possible constraints.

2.2 OT as a weighted-constraint method

Although all OT systems share the same core structure, different choices of EVAL lead to different behaviors. In IBPOT, we use the log-linear EVAL developed by Goldwater and Johnson (2003) in their MaxEnt OT system. MEOT extends traditional OT to account for variation (cases in which multiple candidates can be the winner), as well as gradient/probabilistic productions (Anttila, 1997) and other constraint interactions (e.g., cumulativity) that traditional OT cannot handle (Keller, 2000). MEOT also is motivated by the general MaxEnt framework, whereas most other OT formulations are ad hoc constructions specific to phonology.

In MEOT, each constraint C_i is associated with a weight $w_i < 0$. (Weights are always negative in OT; a constraint violation can never make a candidate more likely to win.) For a given input-candidate pair (x, y), $f_i(y, x)$ is the number of violations of constraint C_i by the pair. As a maximum entropy model, the probability of y given x is proportional to the exponential of the weighted sum of violations, $\sum_i w_i f_i(y, x)$. If $\mathcal{Y}(x)$ is the set of all output candidates for the input x, then the probability of y as the winning output is:

$$p(y|x) = \frac{\exp\left(\sum_i w_i f_i(y, x)\right)}{\sum_{z \in \mathcal{Y}(x)} \exp\left(\sum_i w_i f_i(z, x)\right)} \quad (1)$$

This formulation represents a probabilistic extension of the traditional formulation of OT (Prince and Smolensky, 1993). Traditionally, constraints form a strict hierarchy, where a single violation of a high-ranked constraint is worse than any number of violations of lower-ranked constraints. Traditional OT is also deterministic, with the optimal candidate always selected. In MEOT, the constraint weights define hierarchies of varying strictness, and some probability is assigned to all candidates. If constraints' weights are close together, multiple violations of lower-weighted constraints can reduce a candidate's probability below that of a competitor with a single high-weight violation. As the distance between weights in MEOT increases, the probability of a suboptimal candidate being chosen approaches zero; thus the traditional formulation is a limit case of MEOT.

2.3 OT in practice

Figure 1 shows *tableaux*, a visualization for OT, applied in Wolof (Archangeli and Pulleyblank, 1994; Boersma, 1999). We are interested in four Wolof constraints that combine to induce vowel harmony: *ɪ, PARSE[rtr], HARMONY, and PARSE[atr]. The meaning of these constraints will be discussed in Sect. 4.1; for now, we will only consider their violation profiles. Each column represents a constraint, with weights decreasing left-to-right. Each tableau looks at a single input form, noted in the top-left cell: ete, ɛtɛ, ɪte, or itɛ.

Each row is a candidate output form. A black cell indicates that the candidate, or input-candidate pair, violates the constraint in that column.[2] A white cell indicates no violation. Grey stripes are

[1] Although phonology is usually framed in terms of sound, sign languages also have components that serve equivalent roles in the physical realization of signs (Stokoe, 1960).

[2] In general, a constraint can be violated multiple times by a given candidate, but we will be using binary constraints (violated or not) in this work. See Sect. 5.2 for further discussion.

ete	*ɪ	Parse(rtr)	Harmony	Parse(atr)	Score
ete					0
ɛte			■	▨	-24
etɛ			■	▨	-24
ɛtɛ				■	-8

ɪte	*ɪ	Parse(rtr)	Harmony	Parse(atr)	Score
ite		■			-32
ɪte	■	▨	▨	▨	-80
itɛ		■	■	▨	-56
ɪtɛ	■	▨	▨	▨	-72

ɛtɛ	*ɪ	Parse(rtr)	Harmony	Parse(atr)	Score
ete		■	▨	▨	-32
ɛte		■	▨	▨	-48
etɛ		■	▨	▨	-48
ɛtɛ			▨	▨	0

itɛ	*ɪ	Parse(rtr)	Harmony	Parse(atr)	Score
ite		■	▨	▨	-32
ɪte	■	▨	▨	▨	-120
itɛ			▨	▨	-16
ɪtɛ	■	▨	▨	▨	-72

Figure 1: Tableaux for the Wolof input forms ete, ɛtɛ, ɪte, and itɛ. Black indicates violation, white no violation. Scores are calculated for a MaxEnt OT system with constraint weights of -64, -32, -16, and -8, approximating a traditional hierarchical OT design. Values of grey-striped cells have negligible effects on the distribution (see Sect. 4.3).

overlaid on cells whose value will have a negligible impact on the distribution due to the values of higher-ranked constraint.

Constraints fall into two categories, faithfulness and markedness, which differ in what information they use to assign violations. Faithfulness constraints penalize mismatches between the input and output, while markedness constraints consider only the output. Faithfulness violations include phoneme additions or deletions between the input and output; markedness violations include penalizing specific phonemes in the output form, regardless of whether the phoneme is present in the input.

In MaxEnt OT, each constraint has a weight, and the candidates' scores are the sums of the weights of violated constraints. In the ete tableau at top left, output ete has no violations, and therefore a score of zero. Outputs ɛte and etɛ violate both HARMONY (weight 16) and PARSE[atr] (weight 8), so their scores are 24. Output ɛtɛ violates PARSE[atr], and has score 8. Thus the log-probability of output ɛtɛ is 1/8 that of ete, and the log-probability of disharmonious ɛte and etɛ are each 1/24 that of ete. As the ratio between scores increases, the log-probability ratios can become arbitrarily close to zero, approximating the deterministic situation of traditional OT.

2.4 Learning Constraints

Choosing a winning candidate presumes that a set of constraints CON is available, but where do these constraints come from? The standard assumption within OT is that CON is innate and universal. But in the absence of direct evidence of innate constraints, we should prefer a method that can derive the constraints from cognitively-general learning over one that assumes they are pre-specified. Learning appropriate model features has been an important idea in the development of constraint-based models (Della Pietra et al., 1997).

The innateness assumption can induce tractability issues as well. The strictest formulation of innateness posits that virtually all constraints are shared across all languages, even when there is no evidence for the constraint in a particular language (Tesar and Smolensky, 2000). Strict universality is undermined by the extremely large set of constraints it must weight, as well as the possible existence of language-particular constraints (Smith, 2004).

A looser version of universality supposes that constraints are built compositionally from a set of constraint templates or primitives or phonological features (Hayes, 1999; Smith, 2004; Idsardi, 2006; Riggle, 2009). This version allows language-particular constraints, but it comes with a computational cost, as the learner must be able to generate and evaluate possible constraints while learning the language's phonology. Even with relatively simple constraint templates, such as the phonological constraint learner of Hayes and Wilson (2008), the number of possible constraints expands exponentially. Depending on the specific formulation of the constraints, the constraint identification problem may even be NP-hard (Idsardi, 2006; Heinz et al., 2009). Our approach of casting the learning problem as one of identifying violation profiles is an attempt to determine the amount that can be learned about the active constraints in a paradigm without hypothesizing intensional constraint definitions. The violation profile informa-

tion used by our model could then be used to narrow the search space for intensional constraints, either by performing post-hoc analysis of the constraints identified by our model or by combining intensional constraint search into the learning process. We discuss each of these possibilities in Section 5.2.

Innateness is less of a concern for faithfulness than markedness constraints. Faithfulness violations are determined by the changes between an input form and a candidate, yielding an independent motivation for a universal set of faithfulness constraints (McCarthy, 2008). Some markedness constraints can also be motivated in a universal manner (Hayes, 1999), but many markedness constraints lack such grounding.[3] As such, it is unclear where a universal set of markedness constraints would come from.

3 The IBPOT Model

3.1 Structure

The IBPOT model defines a generative process for mappings between input and output forms based on three latent variables: the constraint violation matrices F (faithfulness) and M (markedness), and the weight vector w. The cells of the violation matrices correspond to the number of violations of a constraint by a given input-output mapping. F_{ijk} is the number of violations of faithfulness constraint F_k by input-output pair type (x_i, y_j); M_{jl} is the number of violations of markedness constraint $M_{\cdot l}$ by output candidate y_j. Note that M is shared across inputs, as M_{jl} has the same value for all input-output pairs with output y_j. The weight vector w provides weight for both F and M. Probabilities of output forms are given by a log-linear function:

$$p(y_j|x_i) = \frac{\exp\left(\sum_k w_k F_{ijk} + \sum_l w_l M_{jl}\right)}{\sum_{y_z \in \mathcal{Y}(x_i)} \exp\left(\sum_k w_k F_{izk} + \sum_l w_l M_{zl}\right)} \quad (2)$$

Note that this is the same structure as Eq. 1 but with faithfulness and markedness constraints listed separately. As discussed in Sect. 2.4, we assume that F is known as part of the output of GEN (Riggle, 2009). The goal of the IBPOT model is to learn the markedness matrix M and weights w for both the markedness and faithfulness constraints.

As for M, we need a non-parametric prior, as there is no inherent limit to the number of markedness constraints a language will use. We use the Indian Buffet Process (Griffiths and Ghahramani, 2005), which defines a proper probability distribution over binary feature matrices with an unbounded number of columns. The IBP can be thought of as representing the set of dishes that diners eat at an infinite buffet table. Each diner (i.e., output form) first draws dishes (i.e., constraint violations) with probability proportional to the number of previous diners who drew it: $p(M_{jl} = 1|\{M_{zl}\}_{z<j}) = n_l/j$. After choosing from the previously taken dishes, the diner can try additional dishes that no previous diner has had. The number of new dishes that the j-th customer draws follows a Poisson(α/j) distribution. The complete specification of the model is then:

$$M \sim IBP(\alpha); \qquad \mathcal{Y}(x_i) = Gen(x_i)$$
$$w \sim -\Gamma(1,1); \quad y|x_i \sim LogLin(M, F, w, \mathcal{Y}(x_i))$$

3.2 Inference

To perform inference in this model, we adopt a common Markov chain Monte Carlo estimation procedure for IBPs (Görür et al., 2006; Navarro and Griffiths, 2007). We alternate approximate Gibbs sampling over the constraint matrix M, using the IBP prior, with a Metropolis-Hastings method to sample constraint weights w.

We initialize the model with a randomly-drawn markedness violation matrix M and weight vector w. To learn, we iterate through the output forms y_j; for each, we split $M_{-j\cdot}$ into "represented" constraints (those that are violated by at least one output form other than y_j) and "non-represented" constraints (those violated only by y_j). For each represented constraint $M_{\cdot l}$, we re-sample the value for the cell M_{jl}. All non-represented constraints are removed, and we propose new constraints, violated only by y_j, to replace them. After each iteration through M, we use Metropolis-Hastings to update the weight vector w.

Represented constraint sampling We begin by resampling M_{jl} for all represented constraints $M_{\cdot l}$, conditioned on the rest of the violations $(M_{-(jl)}, F)$ and the weights w. This is the sampling counterpart of drawing existing features in the IBP generative process. By Bayes' Rule, the

[3] McCarthy (2008, §4.8) gives examples of "ad hoc" intersegmental constraints. Even well-known constraint types, such as generalized alignment, can have disputed structures (Hyde, 2012).

posterior probability of a violation is proportional to product of the likelihood $p(Y|M_{jl} = 1, M_{-jl}, F, w)$ from Eq. 2 and the IBP prior probability $p(M_{jl} = 1|M_{-jl}) = n_{-jl}/n$, where n_{-jl} is the number of outputs other than y_j that violate constraint $M_{\cdot l}$.

Non-represented constraint sampling After sampling the represented constraints for y_j, we consider the addition of new constraints that are violated only by y_j. This is the sampling counterpart to the Poisson draw for new features in the IBP generative process. Ideally, this would draw new constraints from the infinite feature matrix; however, this requires marginalizing the likelihood over possible weights, and we lack an appropriate conjugate prior for doing so. We approximate the infinite matrix with a truncated Bernoulli draw over unrepresented constraints (Görür et al., 2006). We consider in each sample at most K^* new constraints, with weights based on the auxiliary vector w^*. This approximation retains the unbounded feature set of the IBP, as repeated sampling can add more and more constraints without limit.

The auxiliary vector w^* contains the weights of all the constraints that have been removed in the previous step. If the number of constraints removed is less than K^*, w^* is filled out with draws from the prior distribution over weights. We then consider adding any subset of these new constraints to M, each of which would be violated only by y_j. Let M^* represent a (possibly empty) set of constraints paired with a subset of w^*. The posterior probability of drawing M^* from the truncated Bernoulli distribution is the product of the prior probability of M^* $\left(\frac{\frac{\alpha}{K^*}}{N_Y + \frac{\alpha}{K^*}}\right)$ and the likelihood $p(Y|M^*, w^*, M, w, F)$, including the new constraints M^*.

Weight sampling After sampling through all candidates, we use Metropolis-Hastings to estimate new weights for both constraint matrices. Our proposal distribution is $Gamma({w_k}^2/\eta, \eta/w_k)$, with mean w_k and mode $w_k - \frac{\eta}{w_k}$ (for $w_k > 1$). Unlike Gibbs sampling on the constraints, which occurs only on markedness constraints, weights are sampled for both markedness and faithfulness features.

4 Experiment

4.1 Wolof vowel harmony

We test the model by learning the markedness constraints driving Wolof vowel harmony (Archangeli and Pulleyblank, 1994). Vowel harmony in general refers to a phonological phenomenon wherein the vowels of a word share certain features in the output form even if they do not share them in the input. In the case of Wolof, harmony encourages forms that have consistent tongue root positions.

The Wolof vowel system has two relevant features, tongue root position and vowel height. The tongue root can either be advanced (ATR) or retracted (RTR), and the body of the tongue can be in the high, middle, or low part of the mouth. These features define six vowels:

	high	mid	low
ATR	i	e	ə
RTR	ɪ	ɛ	a

We test IBPOT on the harmony system provided in the Praat program (Boersma, 1999), previously used as a test case by Goldwater and Johnson (2003) for MEOT learning with known constraints. This system has four constraints:[4]

- Markedness:
  - *ɪ: do not have ɪ (high RTR vowel)
  - HARMONY: do not have RTR and ATR vowels in the same word
- Faithfulness:
  - PARSE[rtr]: do not change RTR input to ATR output
  - PARSE[atr]: do not change ATR input to RTR output

These constraints define the phonological standard that we will compare IBPOT to, with a ranking from strongest to weakest of *ɪ >> PARSE[rtr] >> HARMONY >> PARSE[atr]. Under this ranking, Wolof harmony is achieved by changing a disharmonious ATR to an RTR, unless this creates an ɪ vowel. We see this in Figure 1, where three of the four winners are harmonic, but with input itɛ, harmony would require violating one of the two higher-ranked constraints. As in previous MEOT work, all Wolof candidates are faithful

[4]The version in Praat includes a fifth constraint, but its value never affects the choice of output in our data and is omitted in this analysis.

with respect to vowel height, either because height changes are not considered by GEN, or because of a high-ranked faithfulness constraint blocking height changes.[5]

The Wolof constraints provide an interesting testing ground for the model, because it is a small set of constraints to be learned, but contains the HARMONY constraint, which can be violated by non-adjacent segments. Non-adjacent constraints are difficult for string-based approaches because of the exponential number of possible relationships across non-adjacent segments. However, the Wolof results show that by learning violations directly, IBPOT does not encounter problems with non-adjacent constraints.

The Wolof data has 36 input forms, each of the form $V_1 t V_2$, where V_1 and V_2 are vowels that agree in height. Each input form has four candidate outputs, with one output always winning. The outputs appear for multiple inputs, as shown in Figure 1. The candidate outputs are the four combinations of tongue-roots for the given vowel heights; the inputs and candidates are known to the learner. We generate simulated data by observing 1000 instances of the winning output for each input.[6] The model must learn the markedness constraints *ɪ and HARMONY, as well as the weights for all four constraints.

We make a small modification to the constraints for the test data: all constraints are limited to binary values. For constraints that can be violated multiple times by an output (e.g., *ɪ twice by ɪtɪ), we use only a single violation. This is necessary in the current model definition because the IBP produces a prior over binary matrices. We generate the simulated data using only single violations of each constraint by each output form. Overcoming the binarity restriction is discussed in Sect. 5.2.

4.2 Experiment Design

We run the model for 10000 iterations, using deterministic annealing through the first 2500 iterations. The model is initialized with a random markedness matrix drawn from the IBP and weights from the exponential prior. We ran versions of the model with parameter settings between 0.01 and 1 for α, 0.05 and 0.5 for η, and 2 and 5 for K^*. All these produced quantitatively similar results; we report values for $\alpha = 1$, $\eta = 0.5$, and $K^* = 5$, which provides the least bias toward small constraint sets.

To establish performance for the phonological standard, we use the IBPOT learner to find constraint weights but do not update M. The resultant learner is essentially MaxEnt OT with the weights estimated through Metropolis sampling instead of gradient ascent. This is done so that the IBPOT weights and phonological standard weights are learned by the same process and can be compared. We use the same parameters for this baseline as for the IBPOT tests. The results in this section are based on nine runs each of IBPOT and MEOT; ten MEOT runs were performed but one failed to converge and was removed from analysis.

4.3 Results

A successful set of learned constraints will satisfy two criteria: achieving good data likelihood (no worse than the phonological-standard constraints) and acquiring constraint violation profiles that are phonologically interpretable. We find that both of these criteria are met by IBPOT on Wolof.

Likelihood comparison First, we calculate the joint probability of the data and model given the priors, $p(Y, M, w|F, \alpha)$, which is proportional to the product of three terms: the data likelihood $p(Y|M, F, w)$, the markedness matrix probability $p(M|\alpha)$, and the weight probability $p(w)$. We present both the mean and MAP values for these over the final 1000 iterations of each run. Results are shown in Table 1.

All eight differences are significant according to t-tests over the nine runs. In all cases but mean M, the IBPOT method has a better log-probability. The most important differences are those in the data probabilities, as the matrix and weight probabilities are reflective primarily of the choice of prior. By both measures, the IBPOT constraints explain the observed data better than the phonologically standard constraints.

Interestingly, the mean M probability is lower for IBPOT than for the phonological standard. Though the phonologically standard constraints

[5] In the present experiment, we assume that GEN does not generate candidates with unfaithful vowel heights. If unfaithful vowel heights were allowed by GEN, these unfaithful candidates would incur a violation approximately as strong as *ɪ, as neither unfaithful-height candidates nor ɪ candidates are attested in the Wolof data.

[6] Since data, matrix, and weight likelihoods all shape the learned constraints, there must be enough data for the model to avoid settling for a simple matrix that poorly explains the data. This represents a similar training set size to previous work (Goldwater and Johnson, 2003; Boersma and Hayes, 2001).

	MAP		Mean	
	IBPOT	PS	IBPOT	PS
Data	-1.52	-3.94	-5.48	-9.23
M	-51.7	-53.3	-54.7	-53.3
w	-44.2	-71.1	-50.6	-78.1
Joint	-97.4	-128.4	-110.6	-140.6

Table 1: Data, markedness matrix, weight vector, and joint log-probabilities for the IBPOT and the phonological standard constraints. MAP and mean estimates over the final 1000 iterations for each run. All IBPOT/PS differences are significant ($p < .005$ for MAP M; $p < .001$ for others).

exist independently of the IBP prior, they fit the prior better than the average IBPOT constraints do. This shows that the IBP's prior preferences can be overcome in order to have constraints that better explain the data.

Constraint comparison Our second criterion is the acquisition of meaningful constraints, that is, ones whose violation profiles have phonologically-grounded explanations. IBPOT learns the same number of markedness constraints as the phonological standard (two); over the final 1000 iterations of the model runs, 99.2% of the iterations had two markedness constraints, and the rest had three.

Turning to the form of these constraints, Figure 2 shows violation profiles from the last iteration of a representative IBPOT run.[7] Because vowel heights must be faithful between input and output, the Wolof data is divided into nine separate *paradigms*, each containing the four candidates (ATR/RTR $\times$ ATR/RTR) for the vowel heights in the input.

The violations on a given output form only affect probabilities within its paradigm. As a result, learned constraints are consistent within paradigms, but across paradigms, the same constraint may serve different purposes.

For instance, the strongest learned markedness constraint, shown as *M1* in Figure 2, has the same violations as the top-ranked constraint that actively distinguishes between candidates in each paradigm. For the five paradigms with at least one high vowel (the top row and left column), *M1* has the same violations as *ɪ, as *ɪ penalizes some but not all of the candidates. In the other four paradigms, *ɪ penalizes none of the candidates, and the IBPOT learner has no reason to learn it. Instead, it learns that *M1* has the same violations as HARMONY, which is the highest-weighted constraint that distinguishes between candidates in these paradigms. Thus in the high-vowel paradigms, *M1* serves as *ɪ, while in the low/mid-vowel paradigms, it serves as HARMONY.

The lower-weighted *M2* is defined noisily, as the higher-ranked *M1* makes some values of *M2* inconsequential. Consider the top-left paradigm of Figure 2, the high-high input, in which only one candidate does not violate *M1* (*ɪ). Because *M1* has a much higher weight than *M2*, a violation of *M2* has a negligible effect on a candidate's probability.[8] In such cells, the constraint's value is influenced more by the prior than by the data. These inconsequential cells are overlaid with grey stripes in Figure 2.

The meaning of *M2*, then, depends only on the consequential cells. In the high-vowel paradigms, *M2* matches HARMONY, and the learned and standard constraints agree on all consequential violations, despite being essentially at chance on the indistinguishable violations (58%). On the non-high paradigms, the meaning of *M2* is unclear, as HARMONY is handled by *M1* and *ɪ is unviolated. In all four paradigms, the model learns that the RTR-RTR candidate violates *M2* and the ATR-ATR candidate does not; this appears to be the model's attempt to reinforce a pattern in the lowest-ranked faithfulness constraint (PARSE[atr]), which the ATR-ATR candidate never violates.

Thus, while the IBPOT constraints are not identical to the phonologically standard ones, they reflect a version of the standard constraints that is consistent with the IBPOT framework.[9] In paradigms where each markedness constraint distinguishes candidates, the learned constraints match the standard constraints. In paradigms where only one constraint distinguishes candidates, the top learned constraint matches it and the second learned constraint exhibits a pattern consistent with a low-ranked faithfulness constraint.

[7] Specifically, from the run with the median joint posterior.

[8] Given the learned weights in Fig. 2, if the losing candidate violates *M1*, its probability changes from 10^{-12} when the preferred candidate does not violate *M2* to 10^{-8} when it does.

[9] In fact, it appears this constraint organization is favored by IBPOT as it allows for lower weights, hence the large difference in w log-probability in Table 1.

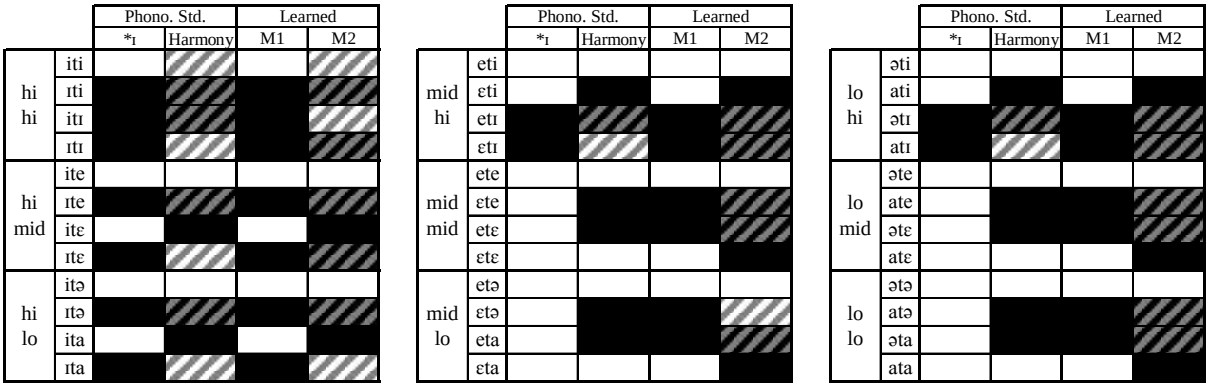


Figure 2: Phonologically standard (*ɪ, HARMONY) and learned (*M1,M2*) constraint violation profiles for the output forms. Learned weights for the standard constraints are -32.8 and -15.3; for *M1* and *M2*, they are -26.5 and -8.4. Black indicates violation, white no violation. Grey stripes indicate cells whose values have negligible effects on the probability distribution.

5 Discussion and Future Work

5.1 Relation to phonotactic learning

Our primary finding from IBPOT is that it is possible to identify constraints that are both effective at explaining the data and representative of theorized phonologically-grounded constraints, given only input-output mappings and faithfulness violations. Furthermore, these constraints are successfully acquired without any knowledge of the phonological structure of the data beyond the faithfulness violation profiles. The model's ability to infer constraint violation profiles without theoretical constraint structure provides an alternative solution to the problems of the traditionally innate and universal OT constraint set.

As it jointly learns constraints and weights, the IBPOT model calls to mind Hayes and Wilson's (2008) joint phonotactic learner. Their learner also jointly learns weights and constraints, but directly selects its constraints from a compositional grammar of constraint definitions. This limits their learner in practice by the rapid explosion in the number of constraints as the maximum constraint definition size grows. By directly learning violation profiles, the IBPOT model avoids this explosion, and the violation profiles can be automatically parsed to identify the constraint definitions that are consistent with the learned profile. The inference method of the two models is different as well; the phonotactic learner selects constraints greedily, whereas the sampling on M in IBPOT asymptotically approaches the posterior.

The two learners also address related but different phonological problems. The phonotactic learner considers phonotactic problems, in which only output matters. The constraints learned by Hayes and Wilson's learner are essentially OT markedness constraints, but their learner does not have to account for varied inputs or effects of faithfulness constraints.

5.2 Extending the learning model

IBPOT, as proposed here, learns constraints based on binary violation profiles, defined extensionally. A complete model of constraint acquisition should provide intensional definitions that are phonologically grounded and cover potentially non-binary constraints. We discuss how to extend the model toward these goals.

IBPOT currently learns extensional constraints, defined by which candidates do or do not violate the constraint. Intensional definitions are needed to extend constraints to unseen forms. Post hoc violation profile analysis, as in Sect. 4.3, provides a first step toward this goal. Such analysis can be integrated into the learning process using the Rational Rules model (Goodman et al., 2008) to identify likely constraint definitions compositionally. Alternately, phonological knowledge could be integrated into a joint constraint learning process in the form of a naturalness bias on the constraint weights or a phonologically-motivated replacement for the IBP prior.

The results presented here use binary constraints, where each candidate violates each constraint only once, a result of the IBP's restriction to binary matrices. Non-binarity can be handled by using the binary matrix M to indicate whether a candidate violates a constraint, with a second

distribution determining the number of violations. Alternately, a binary matrix can directly capture non-binary constraints; Frank and Satta (1998) converted existing non-binary constraints into a binary OT system by representing non-binary constraints as a set of equally-weighted overlapping constraints, each accounting for one violation. The non-binary harmony constraint, for instance, becomes a set {*(at least one disharmony), *(at least two disharmonies), etc.}.

Lastly, the Wolof vowel harmony problem provides a test case with overlaps in the candidate sets for different inputs. This candidate overlap helps the model find appropriate constraint structures. Analyzing other phenomena may require the identification of appropriate abstractions to find this same structural overlap. English regular plurals, for instance, fall into broad categories depending on the features of the stem-final phoneme. IBPOT learning in such settings may require learning an appropriate abstraction as well.

6 Conclusion

A central assumption of Optimality Theory has been the existence of a fixed inventory of universal markedness constraints innately available to the learner, an assumption by arguments regarding the computational complexity of constraint identification. However, our results show for the first time that nonparametric, data-driven learning can identify sparse constraint inventories that both accurately predict the data and are phonologically meaningful, providing a serious alternative to the strong nativist view of the OT constraint inventory.

Acknowledgments

We wish to thank Eric Baković, Emily Morgan, Mark Myslín, the UCSD Computational Psycholinguistics Lab, the Phon Company, and the reviewers for their discussions and feedback on this work. This research was supported by NSF award IIS-0830535 and an Alfred P. Sloan Foundation Research Fellowship to RL.

References

Arto Anttila. 1997. *Variation in Finnish phonology and morphology*. Ph.D. thesis, Stanford U.

Diana Archangeli and Douglas Pulleyblank. 1994. *Grounded phonology*. MIT Press.

Paul Boersma. 1999. Empirical tests of the Gradual Learning Algorithm. *Linguistic Inquiry*, 32:45–86.

Paul Boersma and Bruce Hayes. 2001. Optimality-theoretic learning in the Praat program. In *Proceedings of the Institute of Phonetic Sciences of the University of Amsterdam*.

Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. 1997. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:380–393.

Robert Frank and Giorgio Satta. 1998. Optimality theory and the generative complexity of constraint violability. *Computational Linguistics*, 24:307–315.

Sharon Goldwater and Mark Johnson. 2003. Learning OT constraint rankings using a Maximum Entropy model. In *Proceedings of the Workshop on Variation within Optimality Theory*.

Noah Goodman, Joshua Tenebaum, Jacob Feldman, and Tom Griffiths. 2008. A rational analysis of rule-based concept learning. *Cognitive Science*, 32:108–154.

Dilan Görür, Frank Jäkel, and Carl Rasmussen. 2006. A choice model with infinitely many latent features. In *Proceedings of the 23rd International Conference on Machine Learning*.

Thomas Griffiths and Zoubin Ghahramani. 2005. Infinite latent feature models and the Indian buffet process. Technical Report 2005-001, Gatsby Computational Neuroscience Unit.

Bruce Hayes and Colin Wilson. 2008. A maximum entropy model of phonotactics and phonotactic learning. *Linguistic Inquiry*, 39:379–440.

Bruce Hayes. 1999. Phonetically driven phonology: the role of optimality theory and inductive grounding. In Darnell et al, editor, *Formalism and Functionalism in Linguistics, vol. 1*. Benjamins.

Jeffrey Heinz, Gregory Kobele, and Jason Riggle. 2009. Evaluating the complexity of Optimality Theory. *Linguistic Inquiry*.

Brett Hyde. 2012. Alignment constraints. *Natural Language and Linguistic Theory*, 30:789–836.

William Idsardi. 2006. A simple proof that Optimality Theory is computationally intractable. *Linguistic Inquiry*, 37:271–275.

Frank Keller. 2000. *Gradience in grammar: Experimental and computational aspects of degrees of grammaticality*. Ph.D. thesis, U. of Edinburgh.

John McCarthy. 2008. *Doing Optimality Theory*. Blackwell.

Daniel Navarro and Tom Griffiths. 2007. A nonparametric Bayesian method for inferring features from similarity judgments. In *Advances in Neural Information Processing Systems 19*.

Alan Prince and Paul Smolensky. 1993. Optimality theory: Constraint interaction in generative grammar. Technical report, Rutgers Center for Cognitive Science.

Jason Riggle. 2009. Generating contenders. *Rutgers Optimality Archive*, 1044.

Jennifer Smith. 2004. Making constraints compositional: toward a compositional model of Con. *Lingua*, 114:1433–1464.

William Stokoe. 1960. *Sign Language Structure*. Linstok Press.

Bruce Tesar and Paul Smolensky. 2000. *Learnability in Optimality Theory*. MIT Press.

Active Learning with Efficient Feature Weighting Methods for Improving Data Quality and Classification Accuracy

Justin Martineau[1], Lu Chen[2]*, Doreen Cheng[3], and **Amit Sheth[4]**
[1,3] Samsung Research America, Silicon Valley
[1,3] 75 W Plumeria Dr. San Jose, CA 95134 USA
[2,4] Kno.e.sis Center, Wright State University
[2,4] 3640 Colonel Glenn Hwy. Fairborn, OH 45435 USA
[1,3] {justin.m, doreen.c}@samsung.com
[2,4] {chen, amit}@knoesis.org

Abstract

Many machine learning datasets are noisy with a substantial number of mislabeled instances. This noise yields sub-optimal classification performance. In this paper we study a large, low quality annotated dataset, created quickly and cheaply using Amazon Mechanical Turk to crowdsource annotations. We describe computationally cheap feature weighting techniques and a novel non-linear distribution spreading algorithm that can be used to iteratively and interactively correcting mislabeled instances to significantly improve annotation quality at low cost. Eight different emotion extraction experiments on Twitter data demonstrate that our approach is just as effective as more computationally expensive techniques. Our techniques save a considerable amount of time.

1 Introduction

Supervised classification algorithms require annotated data to teach the machine, by example, how to perform a specific task. There are generally two ways to collect annotations of a dataset: through a few expert annotators, or through crowdsourcing services (e.g., Amazon's Mechanical Turk). High-quality annotations can be produced by expert annotators, but the process is usually slow and costly. The latter option is appealing since it creates a large annotated dataset at low cost. In recent years, there have been an increasing number of studies (Su et al., 2007; Kittur et al., 2008; Sheng et al., 2008; Snow et al., 2008; Callison-Burch, 2009) using crowdsourcing for data annotation. However, because annotators that are recruited this way may lack expertise and motivation, the annotations tend to be more noisy and unreliable, which significantly reduces the performance of the classification model. This is a challenge faced by many real world applications – *given a large, quickly and cheaply created, low quality annotated dataset, how can one improve its quality and learn an accurate classifier from it?*

Re-annotating the whole dataset is too expensive. To reduce the annotation effort, it is desirable to have an algorithm that selects the most likely mislabeled examples first for re-labeling. The process of selecting and re-labeling data points can be conducted with multiple rounds to iteratively improve the data quality. This is similar to the strategy of active learning. The basic idea of active learning is to learn an accurate classifier using less training data. An active learner uses a small set of labeled data to iteratively select the most informative instances from a large pool of unlabeled data for human annotators to label (Settles, 2010). In this work, we borrow the idea of active learning to interactively and iteratively correct labeling errors.

The crucial step is to *effectively and efficiently* select the most likely mislabeled instances. An intuitive idea is to design algorithms that classify the data points and rank them according to the decreasing confidence scores of their labels. The data points with the highest confidence scores but conflicting preliminary labels are most likely mislabeled. The algorithm should be computationally cheap as well as accurate, so it fits well with active learning and other problems that require frequent iterations on large datasets. Specifically, we propose a novel non-linear distribution spreading algorithm, which first uses Delta IDF technique (Martineau and Finin, 2009) to weight features, and then leverages the distribution of Delta IDF scores of a feature across different classes to efficiently recognize discriminative features for the classification task in the presence of mislabeled data. The idea is that some effective fea-

*This author's research was done during an internship with Samsung Research America.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1104–1112,
Baltimore, Maryland, USA, June 23-25 2014.

tures may be subdued due to label noise, and the proposed techniques are capable of counteracting such effect, so that the performance of classification algorithms could be less affected by the noise. With the proposed algorithm, the active learner becomes more accurate and resistant to label noise, thus the mislabeled data points can be more easily and accurately identified.

We consider emotion analysis as an interesting and challenging problem domain of this study, and conduct comprehensive experiments on Twitter data. We employ Amazon's Mechanical Turk (AMT) to label the emotions of Twitter data, and apply the proposed methods to the AMT dataset with the goals of improving the annotation quality at low cost, as well as learning accurate emotion classifiers. Extensive experiments show that, the proposed techniques are as effective as more computational expensive techniques (e.g, Support Vector Machines) but require significantly less time for training/running, which makes it well-suited for active learning.

2 Related Work

Research on handling noisy dataset of mislabeled instances has focused on three major groups of techniques: (1) noise tolerance, (2) noise elimination, and (3) noise correction.

Noise tolerance techniques aim to improve the learning algorithm itself to avoid over-fitting caused by mislabeled instances in the training phase, so that the constructed classifier becomes more noise-tolerant. Decision tree (Mingers, 1989; Vannoorenberghe and Denoeux, 2002) and boosting (Jiang, 2001; Kalaia and Servediob, 2005; Karmaker and Kwek, 2006) are two learning algorithms that have been investigated in many studies. Mingers (1989) explores pruning methods for identifying and removing unreliable branches from a decision tree to reduce the influence of noise. Vannoorenberghe and Denoeux (2002) propose a method based on belief decision trees to handle uncertain labels in the training set. Jiang (2001) studies some theoretical aspects of regression and classification boosting algorithms in dealing with noisy data. Kalaia and Servediob (2005) present a boosting algorithm which can achieve arbitrarily high accuracy in the presence of data noise. Karmaker and Kwek (2006) propose a modified AdaBoost algorithm – ORBoost, which minimizes the impact of outliers and becomes more tolerant to class label noise. One of the main disadvantages of noise tolerance techniques is that they are learning algorithm-dependent. In contrast, noise elimination/correction approaches are more generic and can be more easily applied to various problems.

A large number of studies have explored noise elimination techniques (Brodley and Friedl, 1999; Verbaeten and Van Assche, 2003; Zhu et al., 2003; Muhlenbach et al., 2004; Guan et al., 2011), which identifies and removes mislabeled examples from the dataset as a pre-processing step before building classifiers. One widely used approach (Brodley and Friedl, 1999; Verbaeten and Van Assche, 2003) is to create an ensemble classifier that combines the outputs of multiple classifiers by either majority vote or consensus, and an instance is tagged as mislabeled and removed from the training set if it is classified into a different class than its training label by the ensemble classifier. The similar approach is adopted by Guan et al. (2011) and they further demonstrate that its performance can be significantly improved by utilizing unlabeled data. To deal with the noise in large or distributed datasets, Zhu et al. (2003) propose a partition-based approach, which constructs classification rules from each subset of the dataset, and then evaluates each instance using these rules. Two noise identification schemes, majority and non-objection, are used to combine the decision from each set of rules to decide whether an instance is mislabeled. Muhlenbach et al. (2004) propose a different approach, which represents the proximity between instances in a geometrical neighborhood graph, and an instance is considered suspect if in its neighborhood the proportion of examples of the same class is not significantly greater than in the dataset itself.

Removing mislabeled instances has been demonstrated to be effective in increasing the classification accuracy in prior studies, but there are also some major drawbacks. For example, useful information can be removed with noise elimination, since annotation errors are likely to occur on ambiguous instances that are potentially valuable for learning algorithms. In addition, when the noise ratio is high, there may not be adequate amount of data remaining for building an accurate classifier. The proposed approach does not suffer these limitations.

Instead of eliminating the mislabeled examples

from training data, some researchers (Zeng and Martinez, 2001; Rebbapragada et al., 2012; Laxman et al., 2013) propose to correct labeling errors either with or without consulting human experts. Zeng and Martinez (2001) present an approach based on backpropagation neural networks to automatically correct the mislabeled data. Laxman et al. (2012) propose an algorithm which first trains individual SVM classifiers on several small, class-balanced, random subsets of the dataset, and then reclassifies each training instance using a majority vote of these individual classifiers. However, the automatic correction may introduce new noise to the dataset by mistakenly changing a correct label to a wrong one.

In many scenarios, it is worth the effort and cost to fix the labeling errors by human experts, in order to obtain a high quality dataset that can be reused by the community. Rebbapragada et al. (2012) propose a solution called Active Label Correction (ALC) which iteratively presents the experts with small sets of suspected mislabeled instances at each round. Our work employs a similar framework that uses active learning for data cleaning.

In Active Learning (Settles, 2010) a small set of labeled data is used to find documents that should be annotated from a large pool of unlabeled documents. Many different strategies have been used to select the best points to annotate. These strategies can be generally divided into two groups: (1) selecting points in poorly sampled regions, and (2) selecting points that will have the greatest impact on models that were constructed using the dataset.

Active learning for data cleaning differs from traditional active learning because the data already has low quality labels. It uses the difference between the low quality label for each data point and a prediction of the label using supervised machine learning models built upon the low quality labels. Unlike the work in (Rebbapragada et al., 2012), this paper focuses on developing algorithms that can enhance the ability of active learner on identifying labeling errors, which we consider as a key challenge of this approach but ALC has not addressed.

3 An Active Learning Framework for Label Correction

Let $\hat{D} = \{(x_1, y_1), ..., (x_n, y_n)\}$ be a dataset of binary labeled instances, where the instance x_i belongs to domain X, and its label $y_i \in \{-1, +1\}$. $\hat{D}$ contains an unknown number of mislabeled data points. The problem is to obtain a high-quality dataset D by fixing labeling errors in $\hat{D}$, and learn an accurate classifier C from it.

Algorithm 1 illustrates an active learning approach to the problem. This algorithm takes the noisy dataset $\hat{D}$ as input. The training set T is initialized with the data in $\hat{D}$ and then updated each round with new labels generated during re-annotation. Data sets S_r and S are used to maintain the instances that have been selected for re-annotation in the whole process and in the current iteration, respectively.

Data: noisy data $\hat{D}$
Result: cleaned data D, classifier C
Initialize training set $T = \hat{D}$;
Initialize re-annotated data sets $S_r = \emptyset$;
$S = \emptyset$;
repeat
  Train classifier C using T ;
  Use C to select a set S of m suspected mislabeled instances from T ;
  Experts re-annotate the instances in $S - (S_r \cap S)$;
  Update T with the new labels in S ;
  $S_r = S_r \cup S$; $S = \emptyset$;
until *for I iterations*;
$D = T$;

Algorithm 1: Active Learning Approach for Label Correction

In each iteration, the algorithm trains classifiers using the training data in T. In practice, we apply k-fold cross-validation. We partition T into k subsets, and each time we keep a different subset as testing data and train a classifier using the other $k-1$ subsets of data. This process is repeated k times so that we get a classifier for each of the k subsets. The goal is to use the classifiers to efficiently and accurately seek out the most likely mislabeled instances from T for expert annotators to examine and re-annotate. When applying a classifier to classify the instances in the corresponding data subset, we get the probability about how likely one instance belongs to a class. The top m instances with the highest probabilities belonging to some class but conflicting preliminary labels are selected as the most likely errors for annotators to fix. During the re-annotation process we keep the old labels hidden to prevent that information from

biasing annotators' decisions. Similarly, we keep the probability scores hidden while annotating.

This process is done with multiple iterations of training, sampling, and re-annotating. We maintain the re-annotated instances in S_r to avoid annotating the same instance multiple times. After each round of annotation, we compare the old labels to the new labels to measure the degree of impact this process is having on the dataset. We stop re-annotating on the Ith round after we decide that the reward for an additional round of annotation is too low to justify.

4 Feature Weighting Methods

Building the classifier C that allows the most likely mislabeled instances to be selected and annotated is the essence of the active learning approach. There are two main goals of developing this classifier: (1) accurately predicting the labels of data points and ranking them based on prediction confidence, so that the most likely errors can be effectively identified; (2) requiring less time on training, so that the saved time can be spent on correcting more labeling errors. Thus we aim to build a classifier that is both accurate and time efficient.

Labeling noise affects the classification accuracy. One possible reason is that some effective features that should be given high weights are inhibited in the training phase due to the labeling errors. For example, emoticon ":D" is a good indicator for emotion *happy*, however, if by mistake many instances containing this emoticon are not correctly labeled as *happy*, this class-specific feature would be underestimated during training. Following this idea, we develop computationally cheap feature weighting techniques to counteract such effect by boosting the weight of discriminative features, so that they would not be subdued and the instances with such features would have higher chance to be correctly classified.

Specifically, we propose a non-linear distribution spreading algorithm for feature weighting. This algorithm first utilizes Delta IDF to weigh the features, and then non-linearly spreads out the distribution of features' Delta IDF scores to exaggerate the weight of discriminative features. We first introduce Delta-IDF technique, and then describe our algorithm of distribution spreading. Since we focus on n-gram features, we use the words *feature* and *term* interchangeably in this paper.

4.1 Delta IDF Weighting Scheme

Different from the commonly used TF (term frequency) or TF.IDF (term frequency.inverse document frequency) weighting schemes, Delta IDF treats the positive and negative training instances as two separate corpora, and weighs the terms by how biased they are to one corpus. The more biased a term is to one class, the higher (absolute value of) weight it will get. Delta IDF boosts the importance of terms that tend to be class-specific in the dataset, since they are usually effective features in distinguishing one class from another.

Each training instance (e.g., a document) is represented as a feature vector: $x_i = (w_{1,i}, ..., w_{|V|,i})$, where each dimension in the vector corresponds to a n-gram term in vocabulary $V = \{t_1, ..., t_{|V|}\}$, $|V|$ is the number of unique terms, and $w_{j,i} (1 \leq j \leq |V|)$ is the weight of term t_j in instance x_i. Delta IDF (Martineau and Finin, 2009) assigns score Δ_idf_j to term t_j in V as:

$$\Delta\_idf_j = \log \frac{(N+1)(P_j+1)}{(N_j+1)(P+1)} \quad (1)$$

where P (or N) is the number of positively (or negatively) labeled training instances, P_j (or N_j) is the number of positively (or negatively) labeled training instances with term t_j. Simple add-one smoothing is used to smooth low frequency terms and prevent dividing by zero when a term appears in only one corpus. We calculate the Delta IDF score of every term in V, and get the Delta IDF weight vector $\Delta = (\Delta\_idf_1, ..., \Delta\_idf_{|V|})$ for all terms.

When the dataset is imblanced, to avoid building a biased model, we down sample the majority class before calculating the Delta IDF score and then use the a bias balancing procedure to balance the Delta IDF weight vector. This procedure first divides the Delta IDF weight vector to two vectors, one of which contains all the features with positive scores, and the other of which contains all the features with negative scores. It then applies L2 normalization to each of the two vectors, and add them together to create the final vector.

For each instance, we can calculate the TF.Delta-IDF score as its weight:

$$w_{j,i} = tf_{j,i} \times \Delta\_idf_j \quad (2)$$

where $tf_{j,i}$ is the number of times term t_j occurs in document x_i, and Δ_idf_j is the Delta IDF score of t_j.

4.2 A Non-linear Distribution Spreading Algorithm

Delta IDF technique boosts the weight of features with strong discriminative power. The model's ability to discriminate at the feature level can be further enhanced by leveraging the distribution of feature weights across multiple classes, e.g., multiple emotion categories *funny*, *happy*, *sad*, *exciting*, *boring*, etc.. The distinction of multiple classes can be used to further force feature bias scores apart to improve the identification of class-specific features in the presence of labeling errors.

Let L be a set of target classes, and $|L|$ be the number of classes in L. For each class $l \in L$, we create a binary labeled dataset $\hat{D}^l$. Let V^l be the vocabulary of dataset $\hat{D}^l$, V be the vocabulary of all datasets, and $|V|$ is the number of unique terms in V. Using Formula (1) and dataset $\hat{D}^l$, we get the Delta IDF weight vector for each class l: $\Delta^l = (\Delta\_idf_1^l, ..., \Delta\_idf_{|V|}^l)$. Note that $\Delta\_idf_j^l = 0$ for any term $t_j \in V - V^l$. For a class u, we calculate the spreading score $spread_j^u$ of each feature $t_j \in V$ using a non-linear distribution spreading formula as following (where s is the configurable spread parameter):

$$spread_j^u = \Delta\_idf_j^u \times \frac{\sum_{l \in L-u} |\Delta\_idf_j^u - \Delta\_idf_j^l|^s}{|L| - 1} \quad (3)$$

For any term $t_j \in V$, we can get its Delta IDF score on a class l. The distribution of Delta IDF scores of t_j on all classes in L is represented as $\delta_j = \{\Delta\_idf_j^1, ..., \Delta\_idf_j^{|L|}\}$.

The mechanism of Formula (3) is to non-linearly spread out the distribution, so that the importance of class-specific features can be further boosted to counteract the effect of noisy labels. Specifically, according to Formula (3), a high (absolute value of) spread score indicates that the Delta IDF score of that term on that class is high and deviates greatly from the scores on other classes. In other words, our algorithm assigns high spread score (absolute value) to a term on a class for which the term has strong discriminative power and very specific to that class compared with to other classes. When the dataset is imbalanced, we apply the similar bias balancing procedure as described in Section 4.1 to the spreading model.

While these feature weighting models can be used to score and rank instances for data cleaning, better classification and regression models can be built by using the feature weights generated by these models as a pre-weight on the data points for other machine learning algorithms.

5 Experiments

We conduct experiments on a Twitter dataset that contains tweets about TV shows and movies. The goal is to extract consumers' emotional reactions to multimedia content, which has broad commercial applications including targeted advertising, intelligent search, and recommendation. To create the dataset, we collected 2 billion unique tweets using Twitter API queries for a list of known TV shows and movies on IMDB. Spam tweets were filtered out using a set of heuristics and manually crafted rules. From the set of 2 billion tweets we randomly selected a small subset of 100K tweets about the 60 most highly mentioned TV shows and movies in the dataset. Tweets were randomly sampled for each show using the round robin algorithm. Duplicates were not allowed. This samples an equal number of tweets for each show. We then sent these tweets to Amazon Mechanical Turk for annotation.

We defined our own set of emotions to annotate. The widely accepted emotion taxonomies, including Ekmans Basic Emotions (Ekman, 1999), Russells Circumplex model (Russell and Barrett, 1999), and Plutchiks emotion wheel (Plutchik, 2001), did not fit well for TV shows and Movies. For example, the emotion expressed by laughter is a very important emotion for TV shows and movies, but this emotion is not covered by the taxonomies listed above. After browsing through the raw dataset, reviewing the literature on emotion analysis, and considering the TV and movie problem domain, we decided to focus on eight emotions: *funny*, *happy*, *sad*, *exciting*, *boring*, *angry*, *fear*, and *heartwarming*.

Emotion annotation is a non-trivial task that is typically time-consuming, expensive and error-prone. This task is difficult because: (1) There are multiple emotions to annotate. In this work, we annotate eight different emotions. (2) Emotion expressions could be subtle and ambiguous and thus are easy to miss when labeling quickly. (3) The dataset is very imbalanced, which increases the problem of confirmation bias. As minority classes, emotional tweets can be easily missed because the last X tweets are all not emotional, and the annota-

	Funny	Happy	Sad	Exciting	Boring	Angry	Fear	Heartwarming
# Pos.	1,324	405	618	313	209	92	164	24
# Neg.	88,782	95,639	84,212	79,902	82,443	57,326	46,746	15,857
# Total	90,106	96,044	84,830	80,215	82,652	57,418	46,910	15,881

Table 1: Amazon Mechanical Turk annotation label counts.

	Funny	Happy	Sad	Exciting	Boring	Angry	Fear	Heartwarming
# Pos.	1,781	4,847	788	1,613	216	763	285	326
# Neg.	88,277	91,075	84,031	78,573	82,416	56,584	46,622	15,542
# Total[1]	90,058	95,922	84,819	80,186	82,632	57,347	46,907	15,868

Table 2: Ground truth annotation label counts for each emotion.[2]

tors do not expect the next one to be either. Due to these reasons, there is a lack of sufficient and high quality labeled data for emotion research. Some researchers have studied harnessing Twitter hashtags to automatically create an emotion annotated dataset (Wang et al., 2012).

In order to evaluate our approach in real world scenarios, instead of creating a high quality annotated dataset and then introducing artificial noise, we followed the common practice of crowdsoucing, and collected emotion annotations through Amazon Mechanical Turk (AMT). This AMT annotated dataset was used as the low quality dataset $\hat{D}$ in our evaluation. After that, the same dataset was annotated independently by a group of expert annotators to create the ground truth. We evaluate the proposed approach on two factors, the effectiveness of the models for emotion classification, and the improvement of annotation quality provided by the active learning procedure. We first describe the AMT annotation and ground truth annotation, and then discuss the baselines and experimental results.

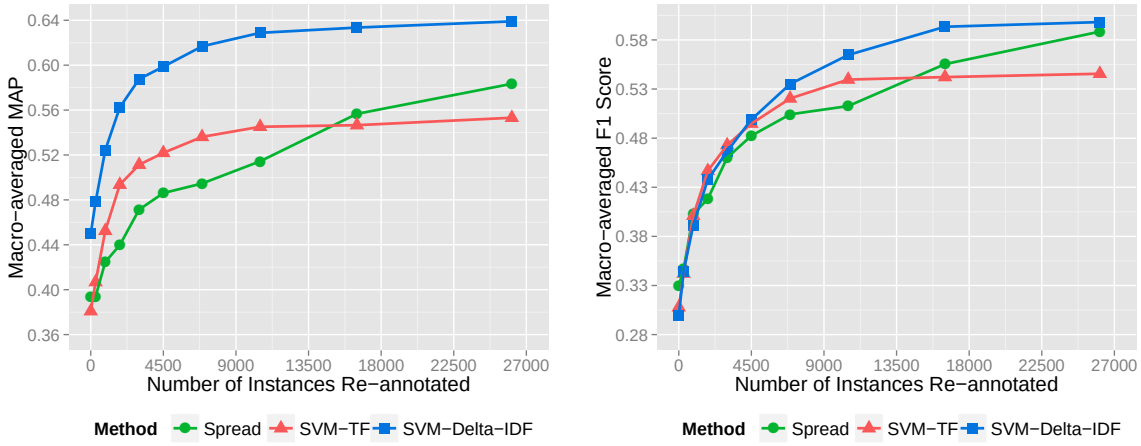
Amazon Mechanical Turk Annotation: we posted the set of 100K tweets to the workers on AMT for emotion annotation. We defined a set of annotation guidelines, which specified rules and examples to help annotators determine when to tag a tweet with an emotion. We applied substantial quality control to our AMT workers to improve the initial quality of annotation following the common practice of crowdsourcing. Each tweet was annotated by at least two workers. We used a series of tests to identify bad workers. These tests include (1) identifying workers with poor pairwise agreement, (2) identifying workers with poor performance on English language annotation, (3) identifying workers that were annotating at unrealistic speeds, (4) identifying workers with near random annotation distributions, and (5) identifying workers that annotate each tweet for a given TV show the same (or nearly the same) way. We manually inspected any worker with low performance on any of these tests before we made a final decision about using any of their annotations.

For further quality control, we also gathered additional annotations from additional workers for tweets where only one out of two workers identified an emotion. After these quality control steps we defined minimum emotion annotation thresholds to determine and assign preliminary emotion labels to tweets. Note that some tweets were discarded as mixed examples for each emotion based upon thresholds for how many times they were tagged, and it resulted in different number of tweets in each emotion dataset. See Table 1 for the statistics of the annotations collected from AMT.

Ground Truth Annotation: After we obtained the annotated dataset from AMT, we posted the same dataset (without the labels) to a group of expert annotators. The experts followed the same annotation guidelines, and each tweet was labeled by at least two experts. When there was a disagreement between two experts, they discussed to reach an agreement or gathered additional opinion from another expert to decide the label of a tweet. We used this annotated dataset as ground truth. See Table 2 for the statistics of the ground truth annotations. Compared with the ground truth, many emotion bearing tweets were missed by the AMT annotators, despite the quality control we applied. It demonstrates the challenge of annotation by crowdsourcing. The imbalanced class distribution

[1]The total number of tweets is lower than the AMT dataset because the experts removed some off-topic tweets.

[2]Expert annotators had a Kappa agreement score of 0.639 before meeting to resolve their differences.



(a) Macro-Averaged MAP

(b) Macro-Averaged F1 Score

Figure 1: Performance comparison of mislabeled instance selection methods. Classifiers become more accurate as more instances are re-annotated. Spread achieves comparable performance with SVMs in terms of both MAP and F1 Score.

aggravates the confirmation bias – the minority class examples are especially easy to miss when labeling quickly due to their rare presence in the dataset.

Evaluation Metric: We evaluated the results with both Mean Average Precision (MAP) and F1 Score. Average Precision (AP) is the average of the algorithm's precision at every position in the confidence ranked list of results where a true emotional document has been identified. Thus, AP places extra emphasis on getting the front of the list correct. MAP is the mean of the average precision scores for each ranked list. This is highly desirable for many practical application such as intelligent search, recommendation, and target advertising where users almost never see results that are not at the top of the list. F1 is a widely-used measure of classification accuracy.

Methods: We evaluated the overall performance relative to the common SVM bag of words approach that can be ubiquitously found in text mining literature. We implemented the following four classification methods:

- **Delta-IDF**: Takes the dot product of the Delta IDF weight vector (Formula 1) with the document's term frequency vector.

- **Spread**: Takes the dot product of the distribution spread weight vector (Formula 3) with the document's term frequency vector. For all the experiments, we used spread parameter $s = 2$.

- **SVM-TF**: Uses a bag of words SVM with term frequency weights.

- **SVM-Delta-IDF**: Uses a bag of words SVM classification with TF.Delta-IDF weights (Formula 2) in the feature vectors before training or testing an SVM.

We employed each method to build the active learner C described in Algorithm 1. We used standard bag of *unigram* and *bigram* words representation and *topic-based fold cross validation*. Since in real world applications people are primarily concerned with how well the algorithm will work for new TV shows or movies that may not be included in the training data, we defined a test fold for each TV show or movie in our labeled data set. Each test fold corresponded to a training fold containing all the labeled data from all the other TV shows and movies. We call it topic-based fold cross validation.

We built the SVM classifiers using LIBLINEAR (Fan et al., 2008) and applied its L2-regularized support vector regression model. Based on the dot product or SVM regression scores, we ranked the tweets by how strongly they express the emotion. We selected the top m tweets with the highest dot product or regression scores but conflicting preliminary AMT labels as the suspected mislabeled instances for re-annotation, just as described in Algorithm 1. For the experimental purpose, the re-annotation was done by assigning the ground truth labels to the selected instances. Since the dataset is highly imbalanced, we applied the *under-sampling strategy* when training the classifiers.

Figure 1 compares the performance of different approaches in each iteration after a certain number of potentially mislabeled instances are re-

annotated. The X axis shows the total number of data points that have been examined for each emotion so far till the current iteration (i.e., 300, 900, 1800, 3000, 4500, 6900, 10500, 16500, and 26100). We reported both the macro-averaged MAP (Figure 1a) and the macro-averaged F1 Score (Figure 1b) on eight emotions as the overall performance of three competitive methods – Spread, SVM-Delta-IDF and SVM-TF. We have also conducted experiments using Delta-IDF, but its performance is low and not comparable with the other three methods.

Generally, Figure 1 shows consistent performance gains as more labels are corrected during active learning. In comparison, SVM-Delta-IDF significantly outperforms SVM-TF with respect to both MAP and F1 Score. SVM-TF achieves higher MAP and F1 Score than Spread at the first few iterations, but then it is beat by Spread after 16,500 tweets had been selected and re-annotated till the eighth iteration. Overall, at the end of the active learning process, Spread outperforms SVM-TF by 3.03% the MAP score (and by 4.29% the F1 score), and SVM-Delta-IDF outperforms SVM-TF by 8.59% the MAP score (and by 5.26% the F1 score). Spread achieves a F1 Score of 58.84%, which is quite competitive compared to 59.82% achieved by SVM-Delta-IDF, though SVM-Delta-IDF outperforms Spread with respect to MAP.

Spread and Delta-IDF are superior with respect to the time efficiency. Figure 2 shows the average training time of the four methods on eight emotions. The time spent training SVM-TF classifiers is twice that of SVM-Delta-IDF classifiers, 12 times that of Spread classifiers, and 31 times that of Delta-IDF classifiers. In our experiments, on average, it took 258.8 seconds to train a SVM-TF classifier for one emotion. In comparison, the average training time of a Spread classifier was only 21.4 seconds, and it required almost no parameter tuning. In total, our method Spread saved up to $(258.8 - 21.4) * 9 * 8 = 17092.8$ seconds (4.75 hours) over nine iterations of active learning for all the eight emotions. This is enough time to re-annotate thousands of data points.

The other important quantity to measure is annotation quality. One measure of improvement for annotation quality is the number of mislabeled instances that can be fixed after a certain number of active learning iterations. Better methods can fix more labels with fewer iterations.

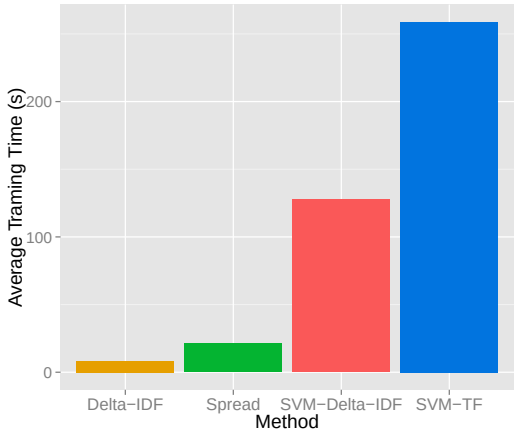


Figure 2: Average training time on eight emotions. Spread requires only one-twelfth of the time spent to training an SVM-TF classifier. Note that the time spent tuning the SVM's parameters has not been included, but is considerable. Compared with such computationally expensive methods, Spread is more appropriate for use with active learning.

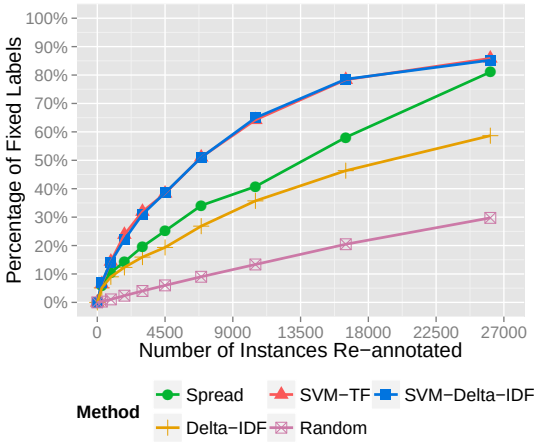


Figure 3: Accumulated average percentage of fixed labels on eight emotions. Spreading the feature weights reduces the number of data points that must be examined in order to correct the mislabeled instances. SVMs require slightly fewer points but take far longer to build.

Besides the four methods, we also implemented a random baseline (Random) which randomly selected the specified number of instances for re-annotation in each round. We compared the improved dataset with the final ground truth at the end of each round to monitor the progress. Figure 3 reports the accumulated average percentage of corrected labels on all emotions in each iteration of the active learning process.

According to the figure, SVM-Delta-IDF and SVM-TF are the most advantageous methods, followed by Spread and Delta-IDF. After the last iteration, SVM-Delta-IDF, SVM-TF, Spread and Delta-IDF has fixed 85.23%, 85.85%, 81.05% and 58.66% of the labels, respectively, all of which significantly outperform the Random baseline (29.74%).

6 Conclusion

In this paper, we explored an active learning approach to improve data annotation quality for classification tasks. Instead of training the active learner using computationally expensive techniques (e.g., SVM-TF), we used a novel non-linear distribution spreading algorithm. This algorithm first weighs the features using the Delta-IDF technique, and then non-linearly spreads out the distribution of the feature scores to enhance the model's ability to discriminate at the feature level. The evaluation shows that our algorithm has the following advantages: (1) It intelligently ordered the data points for annotators to annotate the most likely errors first. The accuracy was at least comparable with computationally expensive baselines (e.g. SVM-TF). (2) The algorithm trained and ran much faster than SVM-TF, allowing annotators to finish more annotations than competitors. (3) The annotation process improved the dataset quality by positively impacting the accuracy of classifiers that were built upon it.

References

Carla E Brodley and Mark A Friedl. 1999. Identifying mislabeled training data. *Journal of Artificial Intelligence Research*, 11:131–167.

Chris Callison-Burch. 2009. Fast, cheap, and creative: evaluating translation quality using amazon's mechanical turk. In *Proceedings of EMNLP*, pages 286–295. ACL.

Paul Ekman. 1999. Basic emotions. *Handbook of cognition and emotion*, 4:5–60.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

Donghai Guan, Weiwei Yuan, Young-Koo Lee, and Sungyoung Lee. 2011. Identifying mislabeled training data with the aid of unlabeled data. *Applied Intelligence*, 35(3):345–358.

Wenxin Jiang. 2001. Some theoretical aspects of boosting in the presence of noisy data. In *Proceedings of ICML*. Citeseer.

Adam Tauman Kalaia and Rocco A Servediob. 2005. Boosting in the presence of noise. *Journal of Computer and System Sciences*, 71:266–290.

Amitava Karmaker and Stephen Kwek. 2006. A boosting approach to remove class label noise. *International Journal of Hybrid Intelligent Systems*, 3(3):169–177.

Aniket Kittur, Ed H Chi, and Bongwon Suh. 2008. Crowdsourcing user studies with mechanical turk. In *Proceedings of CHI*, pages 453–456. ACM.

Srivatsan Laxman, Sushil Mittal, and Ramarathnam Venkatesan. 2013. Error correction in learning using svms. *arXiv preprint arXiv:1301.2012*.

Justin Martineau and Tim Finin. 2009. Delta tfidf: An improved feature space for sentiment analysis. In *Proceedings of ICWSM*.

John Mingers. 1989. An empirical comparison of pruning methods for decision tree induction. *Machine learning*, 4(2):227–243.

Fabrice Muhlenbach, Stéphane Lallich, and Djamel A Zighed. 2004. Identifying and handling mislabelled instances. *Journal of Intelligent Information Systems*, 22(1):89–109.

Robert Plutchik. 2001. The nature of emotions. *American Scientist*, 89(4):344–350.

Umaa Rebbapragada, Carla E Brodley, Damien Sulla-Menashe, and Mark A Friedl. 2012. Active label correction. In *Proceedings of ICDM*, pages 1080–1085. IEEE.

James A Russell and Lisa Feldman Barrett. 1999. Core affect, prototypical emotional episodes, and other things called emotion: dissecting the elephant. *Journal of personality and social psychology*, 76(5):805.

Burr Settles. 2010. Active learning literature survey. *Technical Report 1648, University of Wisconsin, Madison*.

Victor S Sheng, Foster Provost, and Panagiotis G Ipeirotis. 2008. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of KDD*, pages 614–622. ACM.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of EMNLP*, pages 254–263.

Qi Su, Dmitry Pavlov, Jyh-Herng Chow, and Wendell C Baker. 2007. Internet-scale collection of human-reviewed data. In *Proceedings of WWW*, pages 231–240. ACM.

P Vannoorenberghe and T Denoeux. 2002. Handling uncertain labels in multiclass problems using belief decision trees. In *Proceedings of IPMU*, volume 3, pages 1919–1926.

Sofie Verbaeten and Anneleen Van Assche. 2003. Ensemble methods for noise elimination in classification problems. In *Multiple classifier systems*, pages 317–325. Springer.

Wenbo Wang, Lu Chen, Krishnaprasad Thirunarayan, and Amit P Sheth. 2012. Harnessing twitter" big data" for automatic emotion identification. In *Proceedings of SocialCom*, pages 587–592. IEEE.

Xinchuan Zeng and Tony R Martinez. 2001. An algorithm for correcting mislabeled data. *Intelligent data analysis*, 5(6):491–502.

Xingquan Zhu, Xindong Wu, and Qijun Chen. 2003. Eliminating class noise in large datasets. In *Proceedings of ICML*, pages 920–927.

Political Ideology Detection Using Recursive Neural Networks

Mohit Iyyer[1], **Peter Enns**[2], **Jordan Boyd-Graber**[3,4], **Philip Resnik**[2,4]
[1]Computer Science, [2]Linguistics, [3]iSchool, and [4]UMIACS
University of Maryland
{miyyer,peter,jbg}@umiacs.umd.edu, resnik@umd.edu

Abstract

An individual's words often reveal their political ideology. Existing automated techniques to identify ideology from text focus on bags of words or wordlists, ignoring syntax. Taking inspiration from recent work in sentiment analysis that successfully models the compositional aspect of language, we apply a recursive neural network (RNN) framework to the task of identifying the political position evinced by a sentence. To show the importance of modeling subsentential elements, we crowdsource political annotations at a phrase and sentence level. Our model outperforms existing models on our newly annotated dataset and an existing dataset.

1 Introduction

Many of the issues discussed by politicians and the media are so nuanced that even word choice entails choosing an ideological position. For example, what liberals call the "estate tax" conservatives call the "death tax"; there are no ideologically neutral alternatives (Lakoff, 2002). While objectivity remains an important principle of journalistic professionalism, scholars and watchdog groups claim that the media are biased (Groseclose and Milyo, 2005; Gentzkow and Shapiro, 2010; Niven, 2003), backing up their assertions by publishing examples of obviously biased articles on their websites. Whether or not it reflects an underlying lack of objectivity, quantitative changes in the popular framing of an issue over time—favoring one ideologically-based position over another—can have a substantial effect on the evolution of policy (Dardis et al., 2008).

Manually identifying ideological bias in political text, especially in the age of big data, is an impractical and expensive process. Moreover, bias may be localized to a small portion of a document, undetectable by coarse-grained methods. In this paper, we examine the problem of detecting ideological bias on the sentence level. We say a sentence contains *ideological bias* if its author's political position (here *liberal* or *conservative*, in the sense of U.S. politics) is evident from the text.

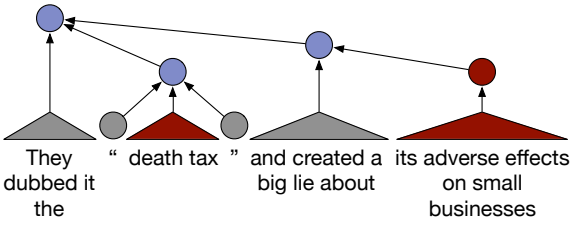


Figure 1: An example of compositionality in ideological bias detection (red → conservative, blue → liberal, gray → neutral) in which modifier phrases and punctuation cause polarity switches at higher levels of the parse tree.

Ideological bias is difficult to detect, even for humans—the task relies not only on political knowledge but also on the annotator's ability to pick up on subtle elements of language use. For example, the sentence in Figure 1 includes phrases typically associated with conservatives, such as "small businesses" and "death tax". When we take more of the structure into account, however, we find that scare quotes and a negative propositional attitude (*a lie about X*) yield an evident liberal bias.

Existing approaches toward bias detection have not gone far beyond "bag of words" classifiers, thus ignoring richer linguistic context of this kind and often operating at the level of whole documents. In contrast, recent work in sentiment analysis has used deep learning to discover compositional effects (Socher et al., 2011b; Socher et al., 2013b).

Building from those insights, we introduce a recursive neural network (RNN) to detect ideological bias on the sentence level. This model requires

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1113–1122,
Baltimore, Maryland, USA, June 23-25 2014.

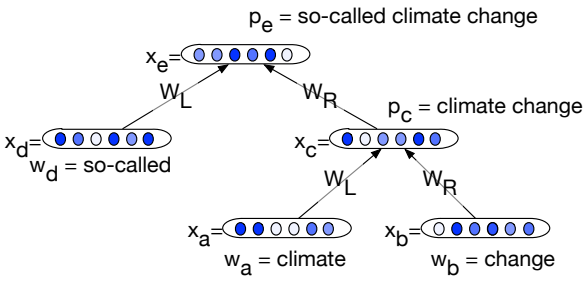


Figure 2: An example RNN for the phrase "so-called climate change". Two d-dimensional word vectors (here, $d = 6$) are composed to generate a phrase vector of the same dimensionality, which can then be recursively used to generate vectors at higher-level nodes.

richer data than currently available, so we develop a new political ideology dataset annotated at the phrase level. With this new dataset we show that RNNs not only label sentences well but also improve further when given additional phrase-level annotations. RNNs are quantitatively more effective than existing methods that use syntactic and semantic features separately, and we also illustrate how our model correctly identifies ideological bias in complex syntactic constructions.

2 Recursive Neural Networks

Recursive neural networks (RNNs) are machine learning models that capture syntactic and semantic composition. They have achieved state-of-the-art performance on a variety of sentence-level NLP tasks, including sentiment analysis, paraphrase detection, and parsing (Socher et al., 2011a; Hermann and Blunsom, 2013). RNN models represent a shift from previous research on ideological bias detection in that they do not rely on hand-made lexicons, dictionaries, or rule sets. In this section, we describe a supervised RNN model for bias detection and highlight differences from previous work in training procedure and initialization.

2.1 Model Description

By taking into account the hierarchical nature of language, RNNs can model *semantic composition*, which is the principle that a phrase's meaning is a combination of the meaning of the words within that phrase and the syntax that combines those words. While semantic composition does not apply universally (e.g., sarcasm and idioms), most language follows this principle. Since most ideological bias becomes identifiable only at higher levels of sentence trees (as verified by our annotation, Figure 4), models relying primarily on word-level distributional statistics are not desirable for our problem.

The basic idea behind the standard RNN model is that each word w in a sentence is associated with a vector representation $x_w \in \mathbb{R}^d$. Based on a parse tree, these words form phrases p (Figure 2). Each of these phrases also has an associated vector $x_p \in \mathbb{R}^d$ of the same dimension as the word vectors. These phrase vectors should represent the meaning of the phrases composed of individual words. As phrases themselves merge into complete sentences, the underlying vector representation is trained to retain the sentence's whole meaning.

The challenge is to describe how vectors combine to form complete representations. If two words w_a and w_b merge to form phrase p, we posit that the phrase-level vector is

$$x_p = f(W_L \cdot x_a + W_R \cdot x_b + b_1), \qquad (1)$$

where W_L and W_R are $d \times d$ left and right composition matrices shared across all nodes in the tree, b_1 is a bias term, and f is a nonlinear activation function such as $\tanh$. The word-level vectors x_a and x_b come from a $d \times V$ dimensional word embedding matrix W_e, where V is the size of the vocabulary.

We are interested in learning representations that can distinguish political polarities given labeled data. If an element of this vector space, x_d, represents a sentence with liberal bias, its vector should be distinct from the vector x_r of a conservative-leaning sentence.

Supervised RNNs achieve this distinction by applying a regression that takes the node's vector x_p as input and produces a prediction $\hat{y}_p$. This is a softmax layer

$$\hat{y}_d = \text{softmax}(W_{cat} \cdot x_p + b_2), \qquad (2)$$

where the softmax function is

$$\text{softmax}(q) = \frac{\exp q}{\sum_{j=1}^{k} \exp q_j} \qquad (3)$$

and W_{cat} is a $k \times d$ matrix for a dataset with k-dimensional labels.

We want the predictions of the softmax layer to match our annotated data; the discrepancy between categorical predictions and annotations is measured

through the cross-entropy loss. We optimize the model parameters to minimize the cross-entropy loss over all sentences in the corpus. The cross-entropy loss of a single sentence is the sum over the true labels y_i in the sentence,

$$\ell(\hat{y}_s) = \sum_{p=1}^{k} y_p * log(\hat{y}_p). \quad (4)$$

This induces a supervised objective function over all sentences: a regularized sum over all node losses normalized by the number of nodes N in the training set,

$$C = \frac{1}{N} \sum_{i}^{N} \ell(pred_i) + \frac{\lambda}{2} \|\theta\|^2. \quad (5)$$

We use L-BFGS with parameter averaging (Hashimoto et al., 2013) to optimize the model parameters $\theta = (W_L, W_R, W_{cat}, W_e, b_1, b_2)$. The gradient of the objective, shown in Eq. (6), is computed using backpropagation through structure (Goller and Kuchler, 1996),

$$\frac{\partial C}{\partial \theta} = \frac{1}{N} \sum_{i}^{N} \frac{\partial \ell(\hat{y}_i)}{\partial \theta} + \lambda\theta. \quad (6)$$

2.2 Initialization

When initializing our model, we have two choices: we can initialize all of our parameters randomly or provide the model some prior knowledge. As we see in Section 4, these choices have a significant effect on final performance.

Random The most straightforward choice is to initialize the word embedding matrix W_e and composition matrices W_L and W_R randomly such that without any training, representations for words and phrases are arbitrarily projected into the vector space.

word2vec The other alternative is to initialize the word embedding matrix W_e with values that reflect the meanings of the associated word types. This improves the performance of RNN models over random initializations (Collobert and Weston, 2008; Socher et al., 2011a). We initialize our model with 300-dimensional *word2vec* toolkit vectors generated by a continuous skip-gram model trained on around 100 billion words from the Google News corpus (Mikolov et al., 2013).

The word2vec embeddings have linear relationships (e.g., the closest vectors to the average of "green" and "energy" include phrases such as "renewable energy", "eco-friendly", and "efficient lightbulbs"). To preserve these relationships as phrases are formed in our sentences, we initialize our left and right composition matrices such that parent vector p is computed by taking the average of children a and b ($W_L = W_R = 0.5\mathbb{I}_{d\times d}$). This initialization of the composition matrices has previously been effective for parsing (Socher et al., 2013a).

3 Datasets

We performed initial experiments on a dataset of Congressional debates that has annotations on the author level for partisanship, not ideology. While the two terms are highly correlated (e.g., a member of the Republican party likely agrees with conservative stances on most issues), they are not identical. For example, a moderate Republican might agree with the liberal position on increased gun control but take conservative positions on other issues. To avoid conflating partisanship and ideology we create a new dataset annotated for ideological bias on the sentence and phrase level. In this section we describe our initial dataset (Convote) and explain the procedure we followed for creating our new dataset (IBC).[1]

3.1 Convote

The Convote dataset (Thomas et al., 2006) consists of US Congressional floor debate transcripts from 2005 in which all speakers have been labeled with their political party (Democrat, Republican, or independent). We propagate party labels down from the speaker to all of their individual sentences and map from party label to ideology label (Democrat → liberal, Republican → conservative). This is an expedient choice; in future work we plan to make use of work in political science characterizing candidates' ideological positions empirically based on their behavior (Carroll et al., 2009).

While the Convote dataset has seen widespread use for document-level political classification, we are unaware of similar efforts at the sentence level.

3.1.1 Biased Sentence Selection

The strong correlation between US political parties and political ideologies (Democrats with liberal, Republicans with conservative) lends confidence that this dataset contains a rich mix of ideological

[1] Available at `http://cs.umd.edu/~miyyer/ibc`

statements. However, the raw Convote dataset contains a low percentage of sentences with explicit ideological bias.[2] We therefore use the features in Yano et al. (2010), which correlate with political bias, to select sentences to annotate that have a higher likelihood of containing bias. Their features come from the Linguistic Inquiry and Word Count lexicon (LIWC) (Pennebaker et al., 2001), as well as from lists of "sticky bigrams" (Brown et al., 1992) strongly associated with one party or another (e.g., "illegal aliens" implies conservative, "universal healthcare" implies liberal).

We first extract the subset of sentences that contains any words in the LIWC categories of Negative Emotion, Positive Emotion, Causation, Anger, and Kill verbs.[3] After computing a list of the top 100 sticky bigrams for each category, ranked by log-likelihood ratio, and selecting another subset from the original data that included only sentences containing at least one sticky bigram, we take the union of the two subsets. Finally, we balance the resulting dataset so that it contains an equal number of sentences from Democrats and Republicans, leaving us with a total of 7,816 sentences.

3.2 Ideological Books

In addition to Convote, we use the Ideological Books Corpus (IBC) developed by Gross et al. (2013). This is a collection of books and magazine articles written between 2008 and 2012 by authors with well-known political leanings. Each document in the IBC has been manually labeled with coarse-grained ideologies (right, left, and center) as well as fine-grained ideologies (e.g., religious-right, libertarian-right) by political science experts.

There are over a million sentences in the IBC, most of which have no noticeable political bias. Therefore we use the filtering procedure outlined in Section 3.1.1 to obtain a subset of 55,932 sentences. Compared to our final Convote dataset, an even larger percentage of the IBC sentences exhibit no noticeable political bias.[4] Because our goal is to distinguish between liberal and conservative bias, instead of the more general task of classifying sentences as "neutral" or "biased", we filter the dataset further using DUALIST (Settles, 2011), an active learning tool, to reduce the proportion of neutral sentences in our dataset. To train the DUALIST classifier, we manually assigned class labels of "neutral" or "biased" to 200 sentences, and selected typical partisan unigrams to represent the "biased" class. DUALIST labels 11,555 sentences as politically biased, 5,434 of which come from conservative authors and 6,121 of which come from liberal authors.

3.2.1 Annotating the IBC

For purposes of annotation, we define the task of political ideology detection as identifying, if possible, the political position of a given sentence's author, where position is either *liberal* or *conservative*.[5] We used the Crowdflower crowdsourcing platform (crowdflower.com), which has previously been used for subsentential sentiment annotation (Sayeed et al., 2012), to obtain human annotations of the filtered IBC dataset for political bias on both the sentence and phrase level. While members of the Crowdflower workforce are certainly not experts in political science, our simple task and the ubiquity of political bias allows us to acquire useful annotations.

Crowdflower Task First, we parse the filtered IBC sentences using the Stanford constituency parser (Socher et al., 2013a). Because of the expense of labeling every node in a sentence, we only label one path in each sentence. The process for selecting paths is as follows: first, if any paths contain one of the top-ten partisan unigrams,[6] we select the longest such path; otherwise, we select the path with the most open class constituencies (NP, VP, ADJP). The root node of a sentence is always included in a path.

Our task is shown in Figure 3. Open class constituencies are revealed to the worker incrementally, starting with the NP, VP, or ADJP furthest from the root and progressing up the tree. We choose this design to prevent workers from changing their lower-level phrase annotations after reading the full sentence.

[2] Many sentences in Convote are variations on "I think this is a good/bad bill", and there is also substantial parliamentary boilerplate language.

[3] While Kill verbs are not a category in LIWC, Yano et al. (2010) adopted it from Greene and Resnik (2009) and showed it to be a useful predictor of political bias. It includes words such as "slaughter" and "starve".

[4] This difference can be mainly attributed to a historical topics in the IBC (e.g., the Crusades, American Civil War). In Convote, every sentence is part of a debate about 2005 political policy.

[5] This is a simplification, as the ideological hierarchy in IBC makes clear.

[6] The words that the multinomial naïve Bayes classifier in DUALIST marked as highest probability given a polarity: market, abortion, economy, rich, liberal, tea, economic, taxes, gun, abortion

Filtering the Workforce To ensure our annotators have a basic understanding of US politics, we restrict workers to US IP addresses and require workers manually annotate one node from 60 different "gold " paths annotated by the authors. We select these nodes such that the associated phrase is either obviously biased or obviously neutral. Workers must correctly annotate at least six of eight gold paths before they are granted access to the full task. In addition, workers must maintain 75% accuracy on gold paths that randomly appear alongside normal paths. Gold paths dramatically improve the quality of our workforce: 60% of contributors passed the initial quiz (the 40% that failed were barred from working on the task), while only 10% of workers who passed the quiz were kicked out for mislabeling subsequent gold paths.

Annotation Results Workers receive the following instructions:

> Each task on this page contains a set of phrases from a single sentence. For each phrase, decide whether or not the author favors a political position to the left (*Liberal*) or right (*Conservative*) of center.
>
> - If the phrase is indicative of a position to the left of center, please choose *Liberal*.
> - If the phrase is indicative of a position to the right of center, please choose *Conservative*.
> - If you feel like the phrase indicates some position to the left or right of the political center, but you're not sure which direction, please mark *Not neutral, but I'm unsure of which direction*.
> - If the phrase is not indicative of a position to the left or right of center, please mark *Neutral*.

We had workers annotate 7,000 randomly selected paths from the filtered IBC dataset, with half of the paths coming from conservative authors and the other half from liberal authors, as annotated by Gross et al. (2013). Three workers annotated each path in the dataset, and we paid $0.03 per sentence. Since identifying political bias is a relatively difficult and subjective task, we include all sentences where at least two workers agree on a label for the root node in our final dataset, except when that label is "Not neutral, but I'm unsure of which direction". We only keep phrase-level annotations where at least two workers agree on the label: 70.4% of all annotated nodes fit this definition of agreement. All unannotated nodes receive the label of their closest annotated ancestor. Since the root of each sentence is always annotated, this strategy ensures that every node in the tree has a label. Our final balanced IBC dataset consists of 3,412 sentences (4,062 before balancing and removing neutral sentences) with a total of 13,640 annotated nodes. Of these sentences, 543 switch polarity (liberal → conservative or vice versa) on an annotated path.

Figure 3: Example political ideology annotation task showing incremental reveal of progressively longer phrases.

While we initially wanted to incorporate neutral labels into our model, we observed that lower-level phrases are almost always neutral while full sentences are much more likely to be biased (Figure 4). Due to this discrepancy, the objective function in Eq. (5) was minimized by making neutral predictions for almost every node in the dataset.

4 Experiments

In this section we describe our experimental framework. We discuss strong baselines that use lexical and syntactic information (including framing-specific features from previous work) as well as multiple RNN configurations. Each of these models have the same task: to predict sentence-level ideology labels for sentences in a test set. To account for label imbalance, we subsample the data so that there are an equal number of labels and report accuracy over this balanced dataset.

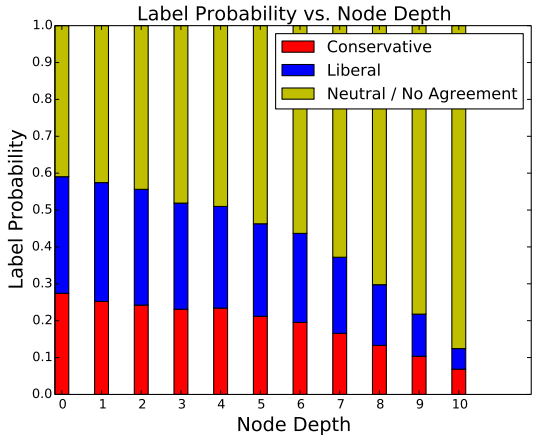


Figure 4: Proportion of liberal, conservative, and neutral annotations with respect to node depth (distance from root). As we get farther from the root of the tree, nodes are more likely to be neutral.

4.1 Baselines

- The **RANDOM** baseline chooses a label at random from {*liberal*, *conservative*}.
- **LR1**, our most basic logistic regression baseline, uses only bag of words (*BoW*) features.
- **LR2** uses only *BoW* features. However, **LR2** also includes phrase-level annotations as separate training instances.[7]
- **LR3** uses *BoW* features as well as syntactic pseudo-word features from Greene & Resnik (2009). These features from dependency relations specify properties of verbs (e.g., transitivity or nominalization).[8]
- **LR-(W2V)** is a logistic regression model trained on the average of the pretrained word embeddings for each sentence (Section 2.2).

The **LR-(W2V)** baseline allows us to compare against a strong lexical representation that encodes syntactic and semantic information without the RNN tree structure. (**LR1**, **LR2**) offer a comparison to simple bag of words models, while the **LR3** baseline contrasts traditional syntactic features with those learned by RNN models.

4.2 RNN Models

For RNN models, we generate a feature vector for every node in the tree. Equation 1 allows us to percolate the representations to the root of the tree. We generate the final instance representation by concatenating the root vector and the average of all other vectors (Socher et al., 2011b). We train an L_2-regularized logistic regression model over these concatenated vectors to obtain final accuracy numbers on the sentence level.

To analyze the effects of initialization and phrase-level annotations, we report results for three different RNN settings. All three models were implemented as described in Section 2 with the nonlinearity f set to the normalized $\tanh$ function,

$$f(v) = \frac{\tanh(v)}{\|\tanh(v)\|}. \quad (7)$$

We perform 10-fold cross-validation on the training data to find the best RNN hyperparameters.[9]

We report results for RNN models with the following configurations:

- **RNN1** initializes all parameters randomly and uses only sentence-level labels for training.
- **RNN1-(W2V)** uses the word2vec initialization described in Section 2.2 but is also trained on only sentence-level labels.
- **RNN2-(W2V)** is initialized using word2vec embeddings and also includes annotated phrase labels in its training. For this model, we also introduce a hyperparameter β that weights the error at annotated nodes $(1-\beta)$ higher than the error at unannotated nodes (β); since we have more confidence in the annotated labels, we want them to contribute more towards the objective function.

Model	Convote	IBC
RANDOM	50%	50%
LR1	64.7%	62.1%
LR2	–	61.9%
LR3	66.9%	62.6%
LR-(W2V)	66.6%	63.7%
RNN1	69.4%	66.2%
RNN1-(W2V)	**70.2%**	67.1%
RNN2-(W2V)	–	**69.3%**

Table 1: Sentence-level bias detection accuracy. The RNN framework, adding phrase-level data, and initializing with word2vec all improve performance over logistic regression baselines. The **LR2** and **RNN2-(W2V)** models were not trained on Convote since it lacks phrase annotations.

[7] The Convote dataset was not annotated on the phrase level, so we only provide a result for the IBC dataset.

[8] We do not include phrase-level annotations in the **LR3** feature set because the pseudo-word features can only be computed from full sentence parses.

[9] $[\lambda_{W_e}$ =1e-6, λ_W =1e-4, $\lambda_{W_{cat}}$ =1e-3, $\beta = 0.3]$

For all RNN models, we set the word vector dimension d to 300 to facilitate direct comparison against the **LR-(W2V)** baseline.[10]

5 Where Compositionality Helps Detect Ideological Bias

In this section, we examine the RNN models to see why they improve over our baselines. We also give examples of sentences that are correctly classified by our best RNN model but incorrectly classified by all of the baselines. Finally, we investigate sentence constructions that our model cannot handle and offer possible explanations for these errors.

Experimental Results Table 1 shows the RNN models outperforming the bag-of-words baselines as well as the word2vec baseline on both datasets. The increased accuracy suggests that the trained RNNs are capable of detecting bias polarity switches at higher levels in parse trees. While phrase-level annotations do not improve baseline performance, the RNN model significantly benefits from these annotations because the phrases are themselves derived from nodes in the network structure. In particular, the phrase annotations allow our best model to detect bias accurately in complex sentences that the baseline models cannot handle.

Initializing the RNN W_e matrix with word2vec embeddings improves accuracy over randomly initialization by 1%. This is similar to improvements from pretrained vectors from neural language models (Socher et al., 2011b).

We obtain better results on Convote than on IBC with both bag-of-words and RNN models. This result was unexpected since the Convote labels are noisier than the annotated IBC labels; however, there are three possible explanations for the discrepancy. First, Convote has twice as many sentences as IBC, and the extra training data might help the model more than IBC's better-quality labels. Second, since the sentences in Convote were originally spoken, they are almost half as short (21.3 words per sentence) as those in the IBC (42.2 words per sentence). Finally, some information is lost at every propagation step, so RNNs are able to model the shorter sentences in Convote more effectively than the longer IBC sentences.

Qualitative Analysis As in previous work (Socher et al., 2011b), we visualize the learned vector space by listing the most probable n-grams for each political affiliation in Table 2. As expected, conservatives emphasize values such as freedom and religion while disparaging excess government spending and their liberal opposition. Meanwhile, liberals inveigh against the gap between the rich and the poor while expressing concern for minority groups and the working class.

Our best model is able to accurately model the compositional effects of bias in sentences with complex syntactic structures. The first three sentences in Figure 5 were correctly classified by our best model (**RNN2-(W2V)**) and incorrectly classified by all of the baselines. Figures 5A and C show traditional conservative phrases, "free market ideology" and "huge amounts of taxpayer money", that switch polarities higher up in the tree when combined with phrases such as "made worse by" and "saved by". Figure 5B shows an example of a bias polarity switch in the opposite direction: the sentence negatively portrays supporters of nationalized health care, which our model picks up on.

Our model often makes errors when polarity switches occur at nodes that are high up in the tree. In Figure 5D, "be used as an instrument to achieve charitable or social ends" reflects a liberal ideology, which the model predicts correctly. However, our model is unable to detect the polarity switch when this phrase is negated with "should not". Since many different issues are discussed in the IBC, it is likely that our dataset has too few examples of some of these issues for the model to adequately learn the appropriate ideological positions, and more training data would resolve many of these errors.

6 Related Work

A growing NLP subfield detects private states such as opinions, sentiment, and beliefs (Wilson et al., 2005; Pang and Lee, 2008) from text. In general, work in this category tends to combine traditional surface lexical modeling (e.g., bag-of-words) with hand-designed syntactic features or lexicons. Here we review the most salient literature related to the present paper.

6.1 Automatic Ideology Detection

Most previous work on ideology detection ignores the syntactic structure of the language in use in favor of familiar bag-of-words representations for

[10] Using smaller vector sizes ($d \in \{50, 100\}$, as in previous work) does not significantly change accuracy.

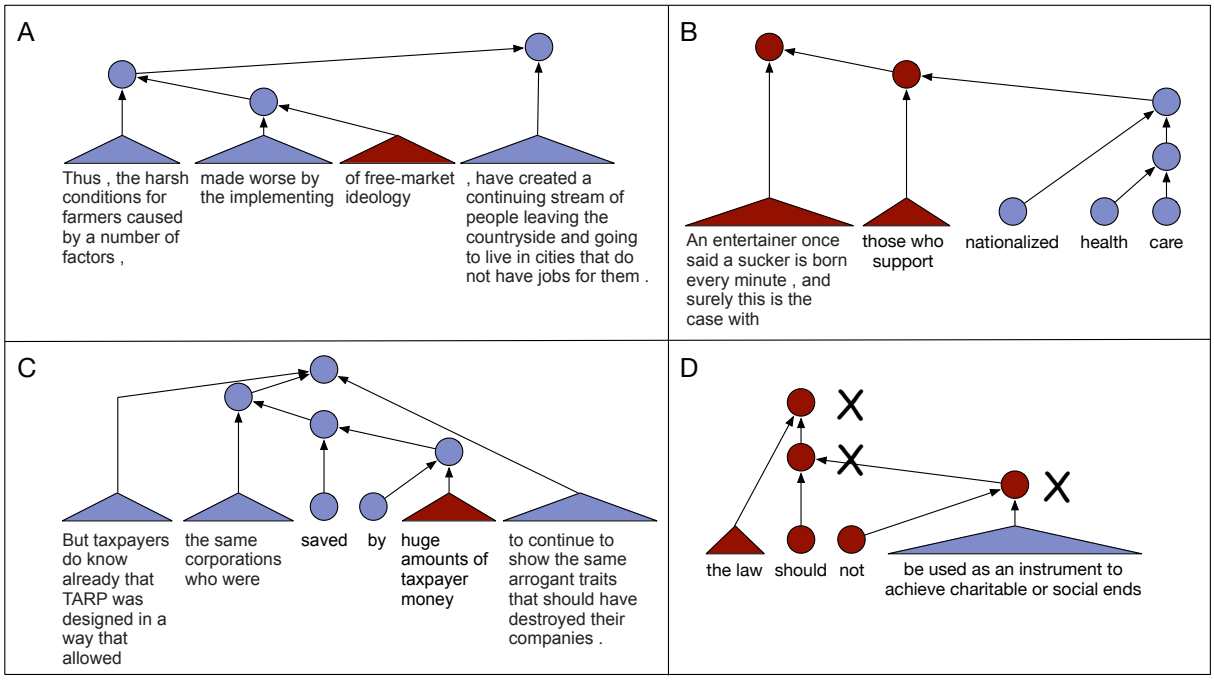


Figure 5: Predictions by **RNN2-(W2V)** on four sentences from the IBC. Node color is the true label (red for conservative, blue for liberal), and an "X" next to a node means the model's prediction was wrong. In *A* and *C*, the model accurately detects conservative-to-liberal polarity switches, while in *B* it correctly predicts the liberal-to-conservative switch. In *D*, negation confuses our model.

the sake of simplicity. For example, Gentzkow and Shapiro (2010) derive a "slant index" to rate the ideological leaning of newspapers. A newspaper's slant index is governed by the frequency of use of partisan collocations of 2-3 tokens. Similarly, authors have relied on simple models of language when leveraging inferred ideological positions. E.g., Gerrish and Blei (2011) predict the voting patterns of Congress members based on bag-of-words representations of bills and inferred political leanings of those members.

Recently, Sim et al. (2013) have proposed a model to infer mixtures of ideological positions in documents, applied to understanding the evolution of ideological rhetoric used by political candidates during the campaign cycle. They use an HMM-based model, defining the states as a set of fine-grained political ideologies, and rely on a closed set of lexical bigram features associated with each ideology, inferred from a manually labeled ideological books corpus. Although it takes elements of discourse structure into account (capturing the"burstiness" of ideological terminology usage), their model explicitly ignores intrasentential contextual influences of the kind seen in Figure 1. Other approaches on the document level use topic models to analyze bias in news articles, blogs, and political speeches (Ahmed and Xing, 2010; Lin et al., 2008; Nguyen et al., 2013).

6.2 Subjectivity Detection

Detecting subjective language, which conveys opinion or speculation, is a related NLP problem. While sentences lacking subjective language may contain ideological bias (e.g., the topic of the sentence), highly-opinionated sentences likely have obvious ideological leanings. In addition, sentiment and subjectivity analysis offers methodological approaches that can be applied to automatic bias detection.

Wiebe et al. (2004) show that low-frequency words and some collocations are a good indicators of subjectivity. More recently, Recasens et al. (2013) detect biased words in sentences using indicator features for bias cues such as hedges and factive verbs in addition to standard bag-of-words and part-of-speech features. They show that this type of linguistic information dramatically improves performance over several standard baselines.

Greene and Resnik (2009) also emphasize the connection between syntactic and semantic relationships in their work on "implicit sentiment",

n	Most conservative n-grams	Most liberal n-grams
1	Salt, Mexico, housework, speculated, consensus, lawyer, pharmaceuticals, ruthless, deadly, Clinton, redistribution	rich, antipsychotic, malaria, biodiversity, richest, gene, pesticides, desertification, Net, wealthiest, labor, fertilizer, nuclear, HIV
3	prize individual liberty, original liberal idiots, stock market crash, God gives freedom, federal government interference, federal oppression nullification, respect individual liberty, Tea Party patriots, radical Sunni Islamists, Obama stimulus programs	rich and poor,"corporate greed", super rich pay, carrying the rich, corporate interest groups, young women workers, the very rich, for the rich, by the rich, soaking the rich, getting rich often, great and rich, the working poor, corporate income tax, the poor migrants
5	spending on popular government programs, bailouts and unfunded government promises, North America from external threats, government regulations place on businesses, strong Church of Christ convictions, radical Islamism and other threats	the rich are really rich, effective forms of worker participation, the pensions of the poor, tax cuts for the rich, the ecological services of biodiversity, poor children and pregnant women, vacation time for overtime pay
7	government intervention helped make the Depression Great, by God in His image and likeness, producing wealth instead of stunting capital creation, the traditional American values of limited government, trillions of dollars to overseas oil producers, its troubled assets to federal sugar daddies, Obama and his party as racialist fanatics	African Americans and other disproportionately poor groups; the growing gap between rich and poor; the Bush tax cuts for the rich; public outrage at corporate and societal greed; sexually transmitted diseases , most notably AIDS; organize unions or fight for better conditions, the biggest hope for health care reform

Table 2: Highest probability n-grams for conservative and liberal ideologies, as predicted by the **RNN2-(W2V)** model.

which refers to sentiment carried by sentence structure and not word choice. They use syntactic dependency relation features combined with lexical information to achieve then state-of-the-art performance on standard sentiment analysis datasets. However, these syntactic features are only computed for a thresholded list of domain-specific verbs. This work extends their insight of modeling sentiment as an interaction between syntax and semantics to ideological bias.

Future Work There are a few obvious directions in which this work can be expanded. First, we can consider more nuanced political ideologies beyond *liberal* and *conservative*. We show that it is possible to detect ideological bias given this binary problem; however, a finer-grained study that also includes neutral annotations may reveal more subtle distinctions between ideologies. While acquiring data with obscure political biases from the IBC or Convote is unfeasible, we can apply a similar analysis to social media (e.g., Twitter or Facebook updates) to discover how many different ideologies propagate in these networks.

Another direction is to implement more sophisticated RNN models (along with more training data) for bias detection. We attempted to apply syntactically-untied RNNs (Socher et al., 2013a) to our data with the idea that associating separate matrices for phrasal categories would improve representations at high-level nodes. While there were too many parameters for this model to work well here, other variations might prove successful, especially with more data. Finally, combining sentence-level and document-level models might improve bias detection at both levels.

7 Conclusion

In this paper we apply recursive neural networks to political ideology detection, a problem where previous work relies heavily on bag-of-words models and hand-designed lexica. We show that our approach detects bias more accurately than existing methods on two different datasets. In addition, we describe an approach to crowdsourcing ideological bias annotations. We use this approach to create a new dataset from the IBC, which is labeled at both the sentence and phrase level.

Acknowledgments

We thank the anonymous reviewers, Hal Daumé, Yuening Hu, Yasuhiro Takayama, and Jyothi Vinjumur for their insightful comments. We also want to thank Justin Gross for providing the IBC and Asad Sayeed for help with the Crowdflower task design, as well as Richard Socher and Karl Moritz Hermann for assisting us with our model implementations. This work was supported by NSF Grant CCF-1018625. Boyd-Graber is also supported by NSF Grant IIS-1320538. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsor.

References

Amr Ahmed and Eric P Xing. 2010. Staying informed: supervised and semi-supervised multi-view topical analysis of ideological perspective. In *EMNLP*.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Comp. Ling.*, 18(4):467–479.

Royce Carroll, Jeffrey B Lewis, James Lo, Keith T Poole, and Howard Rosenthal. 2009. Measuring bias and uncertainty in dw-nominate ideal point estimates via the parametric bootstrap. *Political Analysis*, 17(3):261–275.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*.

Frank E Dardis, Frank R Baumgartner, Amber E Boydstun, Suzanna De Boef, and Fuyuan Shen. 2008. Media framing of capital punishment and its impact on individuals' cognitive responses. *Mass Communication & Society*, 11(2):115–140.

Matthew Gentzkow and Jesse M Shapiro. 2010. What drives media slant? evidence from us daily newspapers. *Econometrica*, 78(1):35–71.

Sean Gerrish and David M Blei. 2011. Predicting legislative roll calls from text. In *ICML*.

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Neural Networks, 1996., IEEE International Conference on*, volume 1.

Stephan Greene and Philip Resnik. 2009. More than words: Syntactic packaging and implicit sentiment. In *NAACL*.

Tim Groseclose and Jeffrey Milyo. 2005. A measure of media bias. *The Quarterly Journal of Economics*, 120(4):1191–1237.

Justin Gross, Brice Acree, Yanchuan Sim, and Noah A Smith. 2013. Testing the etch-a-sketch hypothesis: A computational analysis of mitt romney's ideological makeover during the 2012 primary vs. general elections. In *APSA 2013 Annual Meeting Paper*.

Kazuma Hashimoto, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. 2013. Simple customization of recursive neural networks for semantic relation classification. In *EMNLP*.

Karl Moritz Hermann and Phil Blunsom. 2013. The Role of Syntax in Vector Space Models of Compositional Semantics. In *ACL*.

George Lakoff. 2002. *Moral Politics: How Liberals and Conservatives Think, Second Edition*. University of Chicago Press.

Wei-Hao Lin, Eric Xing, and Alexander Hauptmann. 2008. A joint topic and perspective model for ideological discourse. In *Machine Learning and Knowledge Discovery in Databases*, pages 17–32. Springer.

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

Viet-An Nguyen, Jordan Boyd-Graber, and Philip Resnik. 2013. Lexical and hierarchical topic regression. In *NIPS*, pages 1106–1114.

David Niven. 2003. Objective evidence on media bias: Newspaper coverage of congressional party switchers. *Journalism & Mass Communication Quarterly*, 80(2):311–326.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2).

James W. Pennebaker, Martha E. Francis, and Roger J. Booth. 2001. *Linguistic inquiry and word count [computer software]*. Mahwah, NJ: Erlbaum Publishers.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic models for analyzing and detecting biased language.

Asad B Sayeed, Jordan Boyd-Graber, Bryan Rusk, and Amy Weinberg. 2012. Grammatical structures for word-level sentiment detection. In *NAACL*.

Burr Settles. 2011. Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances. In *EMNLP*.

Yanchuan Sim, Brice Acree, Justin H Gross, and Noah A Smith. 2013. Measuring ideological proportions in political speeches. In *EMNLP*.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011a. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In *NIPS*.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011b. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *EMNLP*.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013a. Parsing With Compositional Vector Grammars. In *ACL*.

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

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from Congressional floor-debate transcripts. In *EMNLP*.

Janyce Wiebe, Theresa Wilson, Rebecca Bruce, Matthew Bell, and Melanie Martin. 2004. Learning subjective language. *Comp. Ling.*, 30(3):277–308.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *EMNLP*.

Tae Yano, Philip Resnik, and Noah A Smith. 2010. Shedding (a thousand points of) light on biased language. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 152–158.

A Unified Model for Soft Linguistic Reordering Constraints in Statistical Machine Translation

Junhui Li[†] **Yuval Marton**[‡] **Philip Resnik**[†] **Hal Daumé III**[†]
[†]UMIACS, University of Maryland, College Park, MD
{lijunhui, resnik, hal}@umiacs.umd.edu
[‡]Microsoft Corp., City Center Plaza, Bellevue, WA
yumarton@microsoft.com

Abstract

This paper explores a simple and effective unified framework for incorporating soft linguistic reordering constraints into a hierarchical phrase-based translation system: 1) a syntactic reordering model that explores reorderings for context free grammar rules; and 2) a semantic reordering model that focuses on the reordering of predicate-argument structures. We develop novel features based on both models and use them as soft constraints to guide the translation process. Experiments on Chinese-English translation show that the reordering approach can significantly improve a state-of-the-art hierarchical phrase-based translation system. However, the gain achieved by the semantic reordering model is limited in the presence of the syntactic reordering model, and we therefore provide a detailed analysis of the behavior differences between the two.

1 Introduction

Reordering models in statistical machine translation (SMT) model the word order difference when translating from one language to another. The popular distortion or lexicalized reordering models in phrase-based SMT make good local predictions by focusing on reordering on word level, while the synchronous context free grammars in hierarchical phrase-based (HPB) translation models are capable of handling non-local reordering on the translation phrase level. However, reordering, especially without any help of external knowledge, remains a great challenge because an accurate reordering is usually beyond these word level or translation phrase level reordering models' ability. In addition, often these translation models fail to respect linguistically-motivated syntax and semantics. As a result, they tend to produce translations containing both syntactic and semantic reordering confusions. In this paper our goal is to take advantage of syntactic and semantic parsing to improve translation quality. Rather than introducing reordering models on either the word level or the translation phrase level, we propose a unified approach to modeling reordering on the linguistic unit level, e.g., syntactic constituents and semantic roles. The reordering unit falls into multiple granularities, from single words to more complex constituents and semantic roles, and often crosses translation phrases. To show the effectiveness of our reordering models, we integrate both syntactic constituent reordering models and semantic role reordering models into a state-of-the-art HPB system (Chiang, 2007; Dyer et al., 2010). We further contrast it with a stronger baseline, already including fine-grained soft syntactic constraint features (Marton and Resnik, 2008; Chiang et al., 2008). The general ideas, however, are applicable to other translation models, e.g., phrase-based model, as well.

Our syntactic constituent reordering model considers context free grammar (CFG) rules in the source language and predicts the reordering of their elements on the target side, using word alignment information. Due to the fact that a constituent, especially a long one, usually maps into multiple discontinuous blocks in the target language, there is more than one way to describe the monotonicity or swapping patterns; we therefore design two reordering models: one is based on the leftmost aligned target word and the other based on the rightmost target word.

While recently there has also been some encouraging work on incorporating semantic structure (or, more specifically, predicate-argument structure: PAS) reordering in SMT, it is still an open question whether semantic structure reordering

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1123–1133,
Baltimore, Maryland, USA, June 23-25 2014.

strongly overlaps with syntactic structure reordering, since the semantic structure is closely tied to syntax. To this end, we employ the same reordering framework as syntactic constituent reordering and focus on semantic roles in a PAS. We then analyze the differences between the syntactic and semantic features.

The contributions of this paper include the following:

- We introduce novel soft reordering constraints, using syntactic constituents or semantic roles, composed over word alignment information in translation rules used during decoding time;
- We introduce a unified framework to incorporate syntactic and semantic reordering constraints;
- We provide a detailed analysis providing insight into why the semantic reordering model is significantly less effective when syntactic reordering features are also present.

The rest of the paper is organized as follows. Section 2 provides an overview of HPB translation model. Section 3 describes the details of our unified reordering models. Section 4 gives our experimental results and Section 5 discusses the behavior difference between syntactic constituent reordering and semantic role reordering. Section 6 reviews related work and, finally Section 7 concludes the paper.

2 HPB Translation Model: an Overview

In HPB models (Chiang, 2007), synchronous rules take the form $X \rightarrow \langle \gamma, \alpha, \sim \rangle$, where X is the non-terminal symbol, γ and α are strings of lexical items and non-terminals in the source and target side, respectively, and $\sim$ indicates the one-to-one correspondence between non-terminals in γ and α. Each such rule is associated with a set of translation model features $\{\phi_i\}$, such as phrase translation probability $p(\alpha \mid \gamma)$ and its inverse $p(\gamma \mid \alpha)$, the lexical translation probability $p_{lex}(\alpha \mid \gamma)$ and its inverse $p_{lex}(\gamma \mid \alpha)$, and a rule penalty that affects preference for longer or shorter derivations. Two other widely used features are a target language model feature and a target word penalty.

Given a derivation d, its translation log-probability is estimated as:

$$\log P(d) \propto \sum_i \lambda_i \phi_i(d) \quad (1)$$

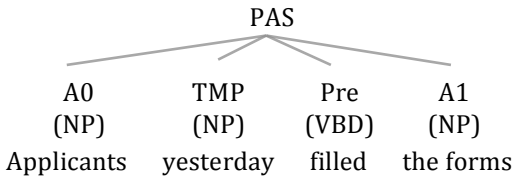


Figure 1: Example of predicate-argument structure.

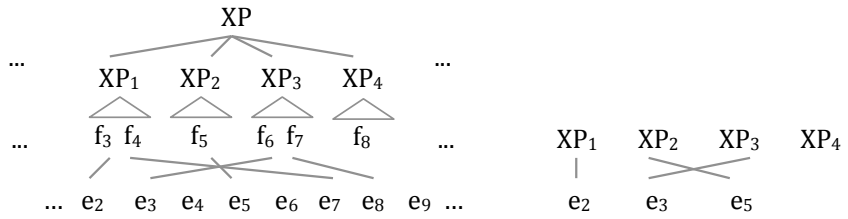
where λ_i is the corresponding weight of feature ϕ_i. See (Chiang, 2007) for more details.

3 Unified Linguistic Reordering Models

As mentioned earlier, the linguistic reordering unit is the syntactic constituent for syntactic reordering, and the semantic role for semantic reordering. The syntactic reordering model takes a CFG rule (e.g., VP $\rightarrow$ VP PP PP) and models the reordering of the constituents on the left hand side by examining their translation or visit order according to the target language. For the semantic reordering model, it takes a PAS and models its reordering on the target side. Figure 1 shows an example of a PAS where the predicate (Pre) has two core arguments (A0 and A1) and one adjunct (TMP). Note that we refer all core arguments, adjuncts, and predicates as semantic roles; thus we say the PAS in Figure 1 has 4 roles. According to the annotation principles in (Chinese) PropBank (Palmer et al., 2005; Xue and Palmer, 2009), all the roles in a PAS map to a corresponding constituent in the parse tree, and these constituents (e.g., NPs and VBD in Figure 1) do not overlap with each other.

Next, we use a CFG rule to describe our syntactic reordering model. Treating the two forms of reorderings in a unified way, the semantic reordering model is obtainable by regarding a PAS as a CFG rule and considering a semantic role as a constituent.

Because the translation of a source constituent might result in multiple discontinuous blocks, there can be several ways to describe or group the reordering patterns. Therefore, we design two general constituent reordering sub-models. One is based on the leftmost aligned word (leftmost reordering model) and the other is based on the rightmost aligned word (rightmost reordering model), as follows. Figure 2 shows the modeling steps for the leftmost reordering model. Figure 2(a) is an example of a CFG rule in the source



(a) a CFG rule and its alignment (b) leftmost aligned target words

XP1	XP2	XP3	XP4
1	4	2	3

(c) visit order

XP1	XP2	XP3	XP4
DM	DS	M	

(d) reordering types

Figure 2: Modeling process illustration for leftmost reordering model.

parse tree and its word alignment links to the target language. Note that constituent XP_4, which covers word f_8, has no alignment. Then for each XP_i, we find the leftmost target word which is aligned to a source word covered by XP_i. Figure 2(b) shows that the leftmost target words for XP_1, XP_2, and XP_3 are e_2, e_5, and e_3, respectively, while XP_4 has no aligned target word. Then we get visit order $V = \{v_i\}$ for $\{XP_i\}$ in the transformation from Figure 2(b) to Figure 2(c), with the following strategies for special cases:

- if the first constituent XP_1 is unaligned, we add a NULL word at the beginning of the target side and link XP_1 to the NULL word;
- if a constituent XP_i $(i > 1)$ is unaligned, we add a link to the target word which is aligned to XP_{i-1}, e.g., XP_4 will be linked to e_3; and
- if k constituents $XP_{m_1} \dots XP_{m_k}$ $(m_1 < \dots < m_k)$ are linked to the same target word, then $v_{m_i} = v_{m_{i+1}} - 1$, e.g., since XP_3 and XP_4 are both linked to e_3, then $v_3 = v_4 - 1$.

Finally Figure 2(d) converts the visit order $V = \{v_1, \dots v_n\}$ into a sequence of leftmost reordering types $LRT = \{lrt_1, \dots, lrt_{n-1}\}$. For every two adjacent constituents XP_i and XP_{i+1} with corresponding visit order v_i and v_{i+1}, their reordering could be one of the following:

- **Monotone (M)** if $v_{i+1} = v_i + 1$;
- **Discontinuous Monotone (DM)** if $v_{i+1} > v_i + 1$;
- **Swap (S)** if $v_{i+1} = v_i - 1$;
- **Discontinuous Swap (DS)** if $v_{i+1} < v_i - 1$.

Up to this point, we have generated a sequence of leftmost reordering types $LRT = \{lrt_1, \dots, lrt_{n-1}\}$ for a given CFG rule *cfg*: $XP \rightarrow XP_1 \dots XP_n$. The leftmost reordering model takes the following form:

$$score_{lrt}(cfg) = P_l(lrt_1, \dots, lrt_{n-1} \mid \psi(cfg)) \quad (2)$$

where $\psi(cfg)$ indicates the surrounding context of the CFG. By assuming that any two reordering types in $LRT = \{lrt_1, \dots, lrt_{n-1}\}$ are independent of each other, we reformulate Eq. 2 into:

$$score_{lrt}(cfg) = \prod_{i=1}^{n-1} P_l(lrt_i \mid \psi(cfg)) \quad (3)$$

Similarly, the sequence of rightmost reordering types *RRT* can be decided for a CFG rule $XP \rightarrow XP_1 \dots XP_n$.

Accordingly, for a PAS *pas*: $PAS \rightarrow R_1 \dots R_n$, we can obtain its sequences of leftmost and rightmost reordering types by using the same way described above.

3.1 Probability Estimation

In order to predict either the leftmost or rightmost reordering type for two adjacent constituents, we use a maximum entropy classifier to estimate the probability of the reordering type $rt \in \{M, DM, S, DS\}$ as follows:

$$P(rt \mid \psi(cfg)) = \frac{exp\left(\sum_k \theta_k f_k(rt, \psi(cfg))\right)}{\sum_{rt'} exp\left(\sum_k \theta_k f_i(rt', \psi(cfg))\right)} \quad (4)$$

where f_k are binary features, θ_k are the weights of these features. Most of our features f_k are syntax-based. For XP_i and XP_{i+1} in *cfg*, the features

#Index	Feature
cf1	$\mathcal{L}(XP_i)$ & $\mathcal{L}(XP_{i+1})$ & $\mathcal{L}(XP)$
cf2	for each XP_j $(j < i)$ $\mathcal{L}(XP_i)$ & $\mathcal{L}(XP_{i+1})$ & $\mathcal{L}(XP)$ & $\mathcal{L}(XP_j)$
cf3	for each XP_j $(j > i+1)$ $\mathcal{L}(XP_i)$ & $\mathcal{L}(XP_{i+1})$ & $\mathcal{L}(XP)$ & $\mathcal{L}(XP_j)$
cf4	$\mathcal{L}(XP_i)$ & $\mathcal{L}(XP_{i+1})$ & $\mathcal{P}(XP_i)$
cf5	$\mathcal{L}(XP_i)$ & $\mathcal{L}(XP_{i+1})$ & $\mathcal{H}(XP_i)$
cf6	$\mathcal{L}(XP_i)$ & $\mathcal{L}(XP_{i+1})$ & $\mathcal{P}(XP_{i+1})$
cf7	$\mathcal{L}(XP_i)$ & $\mathcal{L}(XP_{i+1})$ & $\mathcal{H}(XP_{i+1})$
cf8	$\mathcal{L}(XP_i)$ & $\mathcal{L}(XP_{i+1})$ & $\mathcal{S}(XP_i)$
cf9	$\mathcal{L}(XP_i)$ & $\mathcal{L}(XP_{i+1})$ & $\mathcal{S}(XP_{i+1})$
cf10	$\mathcal{L}(XP_i)$ & $\mathcal{L}(XP)$
cf11	$\mathcal{L}(XP_{i+1})$ & $\mathcal{L}(XP)$

Table 1: Features adopted in the syntactic leftmost and rightmost reordering models. $\mathcal{L}(XP)$ returns the syntactic category of *XP*, e.g., NP, VP, PP etc.; $\mathcal{H}(XP)$ returns the head word of XP; $\mathcal{P}(XP)$ returns the POS tagger of the head word; $\mathcal{S}(XP)$ returns the translation status of *XP* on the target language: *un.* if it is untranslated; *cont.* if it is a continuous block; and *discont.* if it maps into multiple discontinuous blocks.

are aimed to examine which of them should be translated first. Therefore, most features share two common components: the syntactic categories of XP_i and XP_{i+1}. Table 1 shows the features used in syntactic leftmost and rightmost reordering models. Note that we use the same features for both.

Although the semantic reordering model is structured in precisely the same way, we use different feature sets to predict the reordering between two semantic roles. Given the two adjacent roles R_i and R_{i+1} in a PAS *pas*, Table 2 shows the features that are used in the semantic leftmost and rightmost reordering models.

3.2 Integrating into the HPB Model

For models with syntactic reordering, we add two new features (i.e., one for the leftmost reordering model and the other for the rightmost reordering model) into the log-linear translation model in Eq. 1. Unlike the conventional phrase and lexical translation features, whose values are phrase pair-determined and thus can be calculated offline, the value of the reordering features can only be obtained during decoding time, and requires word alignment information as well. Before we present the algorithm integrating the reordering models, we define the following functions by assuming XP_i and XP_{i+1} are the constituent pair of interest in CFG rule *cfg*, H is the translation hypothesis and a is its word alignment:

#Index	Feature
rf1	$\mathcal{R}(R_i)$ & $\mathcal{R}(R_{i+1})$ & $\mathcal{P}(pas)$ $\mathcal{R}(R_i)$ & $\mathcal{R}(R_{i+1})$
rf2	for each R_j $(j < i)$ $\mathcal{R}(R_i)$ & $\mathcal{R}(R_{i+1})$ & $\mathcal{R}(R_j)$ & $\mathcal{P}(pas)$ $\mathcal{R}(R_i)$ & $\mathcal{R}(R_{i+1})$ & $\mathcal{R}(R_j)$
rf3	for each R_j $(j > i+1)$ $\mathcal{R}(R_i)$ & $\mathcal{R}(R_{i+1})$ & $\mathcal{R}(R_j)$ & $\mathcal{P}(pas)$ $\mathcal{R}(R_i)$ & $\mathcal{R}(R_{i+1})$ & $\mathcal{R}(R_j)$
rf4	$\mathcal{R}(R_i)$ & $\mathcal{R}(R_{i+1})$ & $\mathcal{P}(R_i)$
rf5	$\mathcal{R}(R_i)$ & $\mathcal{R}(R_{i+1})$ & $\mathcal{H}(R_i)$
rf6	$\mathcal{R}(R_i)$ & $\mathcal{R}(R_{i+1})$ & $\mathcal{L}(R_i)$
rf7	$\mathcal{R}(R_i)$ & $\mathcal{R}(R_{i+1})$ & $\mathcal{P}(R_{i+1})$
rf8	$\mathcal{R}(R_i)$ & $\mathcal{R}(R_{i+1})$ & $\mathcal{H}(R_{i+1})$
rf9	$\mathcal{R}(R_i)$ & $\mathcal{R}(R_{i+1})$ & $\mathcal{L}(R_{i+1})$
rf10	$\mathcal{R}(R_i)$ & $\mathcal{R}(R_{i+1})$ & $\mathcal{S}(R_i)$
rf11	$\mathcal{R}(R_i)$ & $\mathcal{R}(R_{i+1})$ & $\mathcal{S}(R_{i+1})$
rf12	$\mathcal{R}(R_i)$ & $\mathcal{P}(pas)$ $\mathcal{R}(R_i)$
rf13	$\mathcal{R}(R_{i+1})$ & $\mathcal{P}(pas)$ $\mathcal{R}(R_{i+1})$

Table 2: Features adopted in the semantic leftmost and rightmost reordering models. $\mathcal{P}(pas)$ returns the predicate content of *pas*; $\mathcal{R}(R)$ returns the role type of *R*, e.g., *Pred, A0, TMP,* etc. For features rf1, rf2, rf3, rf12 and rf13, we include another version which excludes the predicate content $\mathcal{P}(pas)$ for reasons of sparsity.

- $\mathcal{F}_1(w_1, w_2, XP)$: returns *true* if constituent *XP* is within the span from word w_1 to w_2; otherwise returns *false*.
- $\mathcal{F}_2(H, cfg, XP_i, XP_{i+1})$ returns *true* if the reordering of the pair $\langle XP_i, XP_{i+1}\rangle$ in rule *cfg* has not been calculated yet; otherwise returns *false*.
- $\mathcal{F}_3(H, a, XP_i, XP_{i+1})$ returns the leftmost and rightmost reordering types for the constituent pair $\langle XP_i, XP_{i+1}\rangle$, given alignment a, according to Section 3.
- $\mathcal{F}_4(rt, cfg, XP_i, XP_{i+1})$ returns the probability of leftmost reordering type rt for the constituent pair $\langle XP_i, XP_{i+1}\rangle$ in rule *cfg*.
- $\mathcal{F}_5(rt, cfg, XP_i, XP_{i+1})$ returns the probability of rightmost reordering type rt for the constituent pair $\langle XP_i, XP_{i+1}\rangle$ in rule *cfg*.

Algorithm 1 integrates the syntactic leftmost and rightmost reordering models into a CKY-style decoder whenever a new hypothesis is generated. Given a hypothesis H with its alignment a, it traverses all CFG rules in the parse tree and sees if two adjacent constituents are conditioned to trigger the reordering models (lines 2-4). For each pair of constituents, it first extracts its leftmost and rightmost reordering types (line 6) and then gets their respective probabilities returned by the maximum entropy classifiers defined in Section 3.1

Algorithm 1: Integrating the syntactic reordering models into a CKY-style decoder

Input: Sentence f in the source language
Parse tree t of f
All CFG rules $\{cfg\}$ in t
Hypothesis H spanning from word w_1 to w_2
Alignment a of H
Output: Log-Probabilities of the syntactic leftmost and rightmost reordering models

1. set $l\_prob = r_prob = 0.0$
2. **foreach** *cfg* in $\{cfg\}$
3. **foreach** pair XP_i and XP_{i+1} in *cfg*
4. **if** $\mathcal{F}_1\,(w_1,\, w_2,\, XP_i)$ = *false* or $\mathcal{F}_1\,(w_1,\, w_2,\, XP_{i+1})$ = *false* or $\mathcal{F}_2\,(H,\, cfg,\, XP_i,\, XP_{i+1})$ = *false*
5. **continue**
6. $(l\_type, r\_type) = \mathcal{F}_3\,(H,\, a,\, XP_i,\, XP_{i+1})$
7. $l\_prob \mathrel{+}= \log \mathcal{F}_4\,(l\_type, cfg, XP_i, XP_{i+1})$
8. $r\_prob \mathrel{+}= \log \mathcal{F}_5\,(r\_type, cfg, XP_i, XP_{i+1})$
9. **return** $(l\_prob,\, r\_prob)$

(lines 7-8). Then the algorithm returns two log-probabilities of the syntactic reordering models. Note that Function $\mathcal{F}_1$ returns true if hypothesis H fully covers, or fully contains, constituent XP_i, regardless of the reordering type of XP_i. Do not confuse any parsing tag XP_i with the nameless variables X_i in Hiero or cdec rules.

For the semantic reordering models, we also add two new features into the log-linear translation model. To get the two semantic reordering model feature values, we simply use Algorithm 1 and its associated functions from $\mathcal{F}_1$ to $\mathcal{F}_5$ replacing a CFG rule *cfg* with a PAS *pas*, and a constituent XP_i with a semantic role R_i. Algorithm 1 therefore permits a unified treatment of syntactic and PAS-based reordering, even though it is expressed in terms of syntactic reordering here for ease of presentation.

4 Experiments

We have presented our unified approach to incorporating syntactic and semantic soft reordering constraints in an HPB system. In this section, we test its effectiveness in Chinese-English translation.

4.1 Experimental Settings

For training we use 1.6M sentence pairs of the non-UN and non-HK Hansards portions of NIST MT training corpora, segmented with the Stanford segmenter (Tseng et al., 2005). The English data is lowercased, tokenized and aligned with GIZA++ (Och and Ney, 2000) to obtain bidirectional alignments, which are symmetrized using the *grow-diag-final-and* method (Koehn et al., 2003). We train a 4-gram LM on the English side of the corpus with 600M additional words from non-NYT and non-LAT, randomly selected portions of the Gigaword v4 corpus, using modified Kneser-Ney smoothing (Chen and Goodman, 1996). We use the HPB decoder cdec (Dyer et al., 2010), with Mr. Mira (Eidelman et al., 2013), which is a k-best variant of MIRA (Chiang et al., 2008), to tune the parameters of the system.

We use NIST MT 06 dataset (1664 sentence pairs) for tuning, and NIST MT 03, 05, and 08 datasets (919, 1082, and 1357 sentence pairs, respectively) for evaluation.[1] We use BLEU (Papineni et al., 2002) for both tuning and evaluation.

To obtain syntactic parse trees and semantic roles on the tuning and test datasets, we first parse the source sentences with the Berkeley Parser (Petrov and Klein, 2007), trained on the Chinese Treebank 7.0 (Xue et al., 2005). We then pass the parses to a Chinese semantic role labeler (Li et al., 2010), trained on the Chinese PropBank 3.0 (Xue and Palmer, 2009), to annotate semantic roles for all verbal predicates (part-of-speech tag *VV*, *VE*, or *VC*).

Our basic baseline system employs 19 basic features: a language model feature, 7 translation model features, word penalty, unknown word penalty, the glue rule, date, number and 6 pass-through features. Our stronger baseline employs, in addition, the fine-grained syntactic soft constraint features of Marton and Resnik (2008), hereafter MR08. The syntactic soft constraint features include both MR08 exact-matching and cross-boundary constraints (denoted *XP=* and *XP+*). Since the syntactic parses of the tuning and test data contain 29 types of constituent labels and 35 types of POS tags, we have 29 types of *XP+* features and 64 types of *XP=* features.

4.2 Model Training

To train the syntactic and semantic reordering models, we use a gold alignment dataset.[2] It contains 7,870 sentences with 191,364 Chinese words and 261,399 English words. We first run syn-

[1] http://www.itl.nist.gov/iad/mig//tests/mt

[2] This dataset includes LDC2006E86, and newswire parts of LDC2012T16, LDC2012T20, LDC2012T24, and LDC2013T05. Indeed, the reordering models can also be trained on the MT training data with its automatic alignment. However, our preliminary experiments showed that the reordering models trained on gold alignment yielded higher improvement.

Reordering Type	Syntactic		Semantic	
	l-m	r-m	l-m	r-m
M	73.5	80.6	63.8	67.9
DM	3.9	3.3	14.0	12.0
S	19.5	13.2	13.1	10.7
DS	3.2	3.0	9.1	9.5
#instance	199,234		66,757	

Table 3: Reordering type distribution over the reordering model's training data. Hereafter, l-m and r-m are for leftmost and rightmost, respectively.

tactic parsing and semantic role labeling on the Chinese sentences, then train the models by using MaxEnt toolkit with L1 regularizer (Tsuruoka et al., 2009).[3] Table 3 shows the reordering type distribution over the training data. Interestingly, about 17% of the syntactic instances and 16% of the semantic instances differ in their leftmost and rightmost reordering types, indicating that the leftmost/rightmost distinction is informative. We also see that the number of semantic instances is about 1/3 of that of syntactic instances, but the entropy of the semantic reordering classes is higher, indicating the reordering of semantic roles is harder than that of syntactic constituents.

A deeper examination of the reordering model's training data reveals that some constituent pairs and semantic role pairs have a preference for a specific reordering type (monotone or swap). In order to understand how well the MR08 system respects their reordering preference, we use the gold alignment dataset LDC2006E86, in which the source sentences are from the Chinese Treebank, and thus both the gold parse trees and gold predicate-argument structures are available. Table 4 presents examples comparing the reordering distribution between gold alignment and the output of the MR08 system. For example, the first row shows that based on the gold alignment, for $\langle PP,VP \rangle$, 16% are in monotone and 76% are in swap reordering. However, our MR08 system outputs 46% of them in monotone and and 50% in swap reordering. Hence, the reordering accuracy for $\langle PP,VP \rangle$ is 54%. Table 4 also shows that the semantic reordering between core arguments and predicates (e.g., $\langle Pred, A1 \rangle$, $\langle A0, Pred \rangle$) has a less ambiguous pattern than that between adjuncts and other roles (e.g., $\langle LOC,Pred \rangle$, $\langle A0,TMP \rangle$), indicating the higher reordering flexibility of adjuncts.

[3] http://www.logos.ic.i.u-tokyo.ac.jp/∼tsuruoka/maxent/

Const. Pair		Gold		MR08 output		
		M	S	M	S	acc.
PP	VP	16	76	46	50	54
NP	LC	26	74	58	42	50
DNP	NP	24	72	78	19	39
CP	NP	26	67	84	10	33
NP	DEG	39	61	31	69	66
...		...		...		
all		81	13	79	14	80
Role Pair		Gold		MR08 output		
		M	S	M	S	acc.
Pred	A1	84	6	82	9	72
A0	Pred	82	11	79	8	75
LOC	Pred	17	30	36	25	49
A0	TMP	35	25	61	6	45
TMP	Pred	30	22	49	19	43
...		...		...		
all		63	13	73	9	64

Table 4: Examples of the reordering distribution (%) of gold alignment and the MR08 system output. For simplicity, we only focus on *(M)onotone* and *(S)wap* based on leftmost reordering.

4.3 Translation Experiment Results

Our first group of experiments investigates whether the syntactic reordering models are able to improve translation quality in terms of BLEU. To this end, we respectively add our syntactic reordering models into both the baseline and MR08 systems. The effect is shown in the rows of "+ syn-reorder" in Table 5. From the table, we have the following two observations.

- Although the HPB model is capable of handling non-local phrase reordering using synchronous context free grammars, both our syntactic leftmost reordering model and rightmost model are still able to achieve improvement over both the baseline and MR08. This suggests that our syntactic reordering features interact well with the MR08 syntactic soft constraints: the XP+ and XP= features focus on a single constituent each, while our reordering features focus on a pair of constituents each.

- There is no clear indication of whether the leftmost reordering model works better than the other. In addition, integrating both the leftmost and rightmost reordering models has limited improvement over a single reordering model.

Our second group of experiments is to validate the semantic reordering models. Results are

System		Tuning	Test			
		MT06	MT03	MT05	MT08	Avg.
Baseline		34.1	36.1	32.3	27.4	31.9
+ syn-reorder	l-m	35.2	36.9‡	33.6‡	28.4‡	33.0
	r-m	35.2	37.2‡	33.7‡	28.6‡	33.2
	both	35.6	37.1‡	33.6‡	28.8‡	33.1
+ sem-reorder	l-m	34.4	36.7‡	33.0‡	27.8†	32.5
	r-m	34.5	36.7‡	33.1‡	27.8‡	32.5
	both	34.5	37.0‡	33.6‡	27.7†	32.8
+syn+sem		35.5	37.3‡	33.7‡	29.0‡	33.3
MR08		35.6	37.4	34.2	28.7	33.4
+ syn-reorder	l-m	36.0	38.2‡	35.0‡	29.2‡	34.1
	r-m	36.0	38.1‡	34.8‡	29.2‡	34.0
	both	35.9	38.2‡	35.3‡	29.5‡	34.3
+ sem-reorder	l-m	35.8	37.6†	34.7‡	28.7	33.7
	r-m	35.8	37.4	34.5†	28.8	33.6
	both	35.8	37.6†	34.7‡	28.8	33.7
+syn+sem		36.1	38.4‡	35.2‡	29.5‡	34.4

Table 5: System performance in BLEU scores. ‡/†: significant over baseline or MR08 at 0.01 / 0.05, respectively, as tested by bootstrap resampling (Koehn, 2004)

shown in the rows of "+ sem-reorder" in Table 5. Here we observe:

- The semantic reordering models also achieve significant gain of 0.8 BLEU on average over the baseline system, demonstrating the effectiveness of PAS-based reordering. However, the gain diminishes to 0.3 BLEU on the MR08 system.
- The syntactic reordering models outperform the semantic reordering models on both the baseline and MR08 systems.

Finally, we integrate both the syntactic and semantic reordering models into the final system. The two models collectively achieve a gain of up to 1.4 BLEU over the baseline and 1.0 BLEU over MR08 on average, which is shown in the rows of "+syn+sem" in Table 5.

5 Discussion

The trend of the results, summarized as performance gain over the baseline and MR08 systems averaged over all test sets, is presented in Table 6. The syntactic reordering models outperform the semantic reordering models, and the gain achieved by the semantic reordering models is limited in the presence of the MR08 syntactic features. In this section, we look at MR08 system and the systems improving it to explore the behavior differences between the two reordering models.

Coverage analysis: Our statistics show that syntactic reordering features (either leftmost or rightmost) are called 24 times per sentence on average. This is compared to only 9 times per sentence for semantic reordering features. This is not surprising since the semantic reordering features are exclusively attached to predicates, and the span set of the semantic roles is a strict subset of the span set of the syntactic constituents; only 22% of syntactic constituents are semantic roles. On average, a sentences has 4 PASs and each PAS contains 3 semantic roles. Of all the semantic role pairs, 44% are in the same CFG rules, indicating that this part of semantic reordering has overlap with syntactic reordering. Therefore, the PAS model has fewer opportunities to influence reordering.

System	Baseline	MR08
+syn-reorder	1.2	0.9
+sem-reorder	0.8	0.3
+ both	1.4	1.0

Table 6: Performance gain in BLEU over baseline and MR08 systems averaged over all test sets.

Reordering accuracy analysis: The reordering type distribution on the reordering model training data in Table 3 suggests that semantic reordering is more difficult than syntactic reordering. To validate this conjecture on our translation test data, we compare the reordering performance among the MR08 system, the improved systems and the maximum entropy classifiers. For the test set, we have four reference translations. We run GIZA++ on the data combination of our translation training data and test data to get the alignment for the test data and each reference translation. Once we have the (semi-)gold alignment, we compute the gold reordering types between two adjacent syntactic constituents or semantic roles. Then we evaluate the automatic reordering outputs generated from both our translation systems and maximum entropy classifiers. Table 7 shows the accuracy averaged over the four gold reordering sets (the four reference translations). It shows that 1) as expected, our classifiers do worse on the harder semantic reordering prediction than syntactic reordering prediction; 2) thanks to the high accuracy obtained by the maxent classifiers, integrating either the syntactic or the semantic reordering constraints results in better reordering performance from both syntactic and semantic perspectives; 3) in terms of the mutual impact, the syntactic reordering models help improving semantic reordering more than the semantic reordering

System	Syntactic		Semantic	
	l-m	r-m	l-m	r-m
MR08	75.0	78.0	66.3	68.5
+syn-reorder	78.4	80.9	69.0	70.2
+sem-reorder	76.0	78.8	70.7	72.7
+both	78.6	81.7	70.6	72.1
Maxent Classifier	80.7	85.6	70.9	73.5

Table 7: Reordering accuracy on four gold sets.

System	Syntactic		Semantic	
	l-m	r-m	l-m	r-m
+syn-reorder	1.2	1.2	-	-
+sem-reorder	-	-	0.7	0.9
+both	1.2	1.0	0.5	0.4

Table 8: Reordering feature weights.

models help improving syntactic reordering; and 4) the rightmost models have a learnability advantage over the leftmost models, achieving higher accuracy across the board.

Feature weight analysis: Table 8 shows the syntactic and semantic reordering feature weights. It shows that the semantic feature weights decrease in the presence of the syntactic features, indicating that the decoder learns to trust semantic features less in the presence of the more accurate syntactic features. This is consistent with our observation that semantic reordering is harder than syntactic reordering, as seen in Tables 3 and 7.

Potential improvement analysis: Table 7 also shows that our current maximum entropy classifiers have room for improvement, especially for semantic reordering. In order to explore the error propagation from the classifiers themselves and explore the upper bound for improvement from the reordering models, we perform an "oracle" study, letting the classifiers be aware of the "gold" reordering type between two syntactic constituents or two semantic roles, and returning a higher probability for the gold reordering type and a smaller one for the others (i.e., we set 0.9 for the gold reordering type, and let the other non-gold three types share 0.1). Again, to get the gold reordering type, we run GIZA++ to get the alignment for tuning/test source sentences and each of four reference translations. We report the averaged performance by using the gold reordering type extracted from the four reference translations. Table 9 compares the performance between the non-oracle and oracle settings. We clearly see that using gold syntactic reordering types significantly improves the performance (e.g., 34.9 vs. 33.4 on average) and there is still some room for improvement by building a better maximum entropy classifiers (e.g., 34.9 vs. 34.3). To our surprise, however, the improvement achieved by gold semantic reordering types is still small (e.g., 33.9 vs. 33.4), suggesting that the potential improvement of semantic reordering models is much more limited. And we again see that the improvement achieved by semantic reordering models is limited in the presence of the syntactic reordering models.

	System	MT 03	MT 05	MT 08	Avg.
Non-Oracle	MR08	37.4	34.2	28.7	33.4
	+syn-reorder	38.2	35.3	29.5	34.3
	+sem-reorder	37.6	34.7	28.8	33.7
	+ both	38.4	35.2	29.5	34.4
Oracle	+syn-reorder	39.2	35.9	29.6	34.9
	+sem-reorder	37.9	34.8	28.9	33.9
	+ both	39.1	36.0	29.8	35.0

Table 9: Performance (BLEU score) comparison between non-oracle and oracle experiments.

6 Related Work

Syntax-based reordering: Some previous work pre-ordered words in the source sentences, so that the word order of source and target sentences is similar. The reordering rules were either manually designed (Collins et al., 2005; Wang et al., 2007; Xu et al., 2009; Lee et al., 2010) or automatically learned (Xia and McCord, 2004; Genzel, 2010; Visweswariah et al., 2010; Khalilov and Sima'an, 2011; Lerner and Petrov, 2013), using syntactic parses. Li et al. (2007) focused on finding the n-best pre-ordered source sentences by predicting the reordering of sibling constituents, while Yang et al. (2012) obtained word order by using a reranking approach to reposition nodes in syntactic parse trees. Both are close to our work; however, our model generates reordering features that are integrated into the log-linear translation model during decoding.

Another approach in previous work added soft constraints as weighted features in the SMT decoder to reward good reorderings and penalize bad ones. Marton and Resnik (2008) employed soft syntactic constraints with weighted binary features and no MaxEnt model. They did not explicitly target reordering (beyond applying constraints on HPB rules). Although employing linguistically motivated labels in SCFG is capable of capturing constituent reorderings (Chiang, 2010; Mylon-

akis and Sima'an, 2011), the rules are sparser than SCFG with nameless non-terminals (i.e., *X*s) and soft constraints. Ge (2010) presented a syntax-driven maximum entropy reordering model that predicted the source word translation order. Gao et al. (2011) employed dependency trees to predict the translation order of a word and its head word. Huang et al. (2013) predicted the translation order of two source words.[4] Our work, which shares this approach, differs from their work primarily in that our syntactic reordering models are based on the constituent level, rather than the word level.

Semantics-based reordering: Semantics-based reordering has also seen an increase in activity recently. In the pre-ordering approach, Wu et al. (2011) automatically learned pre-ordering rules from PAS. In the soft constraint or reordering model approach, Liu and Gildea (2010) modeled the reordering/deletion of source-side semantic roles in a tree-to-string translation model. Xiong et al. (2012) and Li et al. (2013) predicted the translation order between either two arguments or an argument and its predicate. Instead of decomposing a PAS into individual units, Zhai et al. (2013) constructed a classifier for each source side PAS. Finally in the post-processing approach category, Wu and Fung (2009) performed semantic role labeling on translation output and reordered arguments to maximize the cross-lingual match of the semantic frames between the source sentence and the target translation. To our knowledge, their semantic reordering models were PAS-specific. In contrast, our model is universal and can be easily adopted to model the reordering of other linguistic units (e.g., syntactic constituents). Moreover, we have studied the effectiveness of the semantic reordering model in different scenarios.

Non-syntax-based reorderings in HPB: Recently we have also seen work on lexicalized reordering models without syntactic information in HPB (Setiawan et al., 2009; Huck et al., 2013; Nguyen and Vogel, 2013). The non-syntax-based reordering approach models the reordering of translation words/phrases while the syntax-based approach models the reordering of syntactic constituents. Although there are overlaps between translation phrases and syntactic constituents, it is reasonable to think that the two reordering approaches can work together well and even complement each other, as the linguistic patterns they capture differ substantially. Setiawan et al. (2013) modeled the orientation decisions between anchors and two neighboring multi-unit chunks which might cross phrase or rule boundaries. Last, we also note that recent work on non-syntax-based reorderings in (flat) phrase-based models (Cherry, 2013; Feng et al., 2013) can also be potentially adopted to hpb models.

[4]Note that they obtained the translation order of source word pairs by predicting the reordering of adjacent constituents, which was quite close to our work.

7 Conclusion and Future Work

In this paper, we have presented a unified reordering framework to incorporate soft linguistic constraints (of syntactic or semantic nature) into the HPB translation model. The syntactic reordering models take CFG rules and model their reordering on the target side, while the semantic reordering models work with PAS. Experiments on Chinese-English translation show that the reordering approach can significantly improve a state-of-the-art hierarchical phrase-based translation system. We have also discussed the differences between the two linguistic reordering models.

There are many directions in which this work can be continued. First, the syntactic reordering model can be extended to model reordering among constituents that cross CFG rules. Second, although we do not see obvious gain from the semantic reordering model when the syntactic model is adopted, it might be beneficial to further jointly consider the two reordering models, focusing on where each one does well. Third, to better examine the overlap or synergy between our approach and the non-syntax-based reordering approach, we will conduct direct comparisons and combinations with the latter.

Acknowledgments

This research was supported in part by the BOLT program of the Defense Advanced Research Projects Agency, Contract No. HR0012-12-C-0015. Any opinions, findings, conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the view of DARPA. The authors would like to thank three anonymous reviewers for providing helpful comments, and also acknowledge Ke Wu, Vladimir Eidelman, Hua He, Doug Oard, Yuening Hu, Jordan Boyd-Graber, and Jyothi Vinjumur for useful discussions.

References

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of ACL 1996*, pages 310–318.

Colin Cherry. 2013. Improved reordering for phrase-based translation using sparse features. In *Proceedings of HLT-NAACL 2013*, pages 22–31.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proceedings of EMNLP 2008*, pages 224–233.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

David Chiang. 2010. Learning to translate with source and target syntax. In *Proceedings of ACL 2010*, pages 1443–1452.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of ACL 2005*, pages 531–540.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Jonathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of ACL 2010 System Demonstrations*, pages 7–12.

Vladimir Eidelman, Ke Wu, Ferhan Ture, Philip Resnik, and Jimmy Lin. 2013. Mr. mira: Open-source large-margin structured learning on mapreduce. In *Proceedings of ACL 2013 System Demonstrations*, pages 199–204.

Minwei Feng, Jan-Thorsten Peter, and Hermann Ney. 2013. Advancements in reordering models for statistical machine translation. In *Proceedings of ACL 2013*, pages 322–332.

Yang Gao, Philipp Koehn, and Alexandra Birch. 2011. Soft dependency constraints for reordering in hierarchical phrase-based translation. In *Proceedings of EMNLP 2011*, pages 857–868.

Niyu Ge. 2010. A direct syntax-driven reordering model for phrase-based machine translation. In *Proceedings of HLT-NAACL 2010*, pages 849–857.

Dmitriy Genzel. 2010. Automatically learning source-side reordering rules for large scale machine translation. In *Proceedings of COLING 2010*, pages 376–384.

Zhongqiang Huang, Jacob Devlin, and Rabih Zbib. 2013. Factored soft source syntactic constraints for hierarchical machine translation. In *Proceedings of EMNLP 2013*, pages 556–566.

Matthias Huck, Joern Wuebker, Felix Rietig, and Hermann Ney. 2013. A phrase orientation model for hierarchical machine translation. In *Proceedings of WMT 2013*, pages 452–463.

Maxim Khalilov and Khalil Sima'an. 2011. Context-sensitive syntactic source-reordering by statistical transduction. In *Proceedings of IJCNLP 2011*, pages 38–46.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL 2003*, pages 48–54.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP 2004*, pages 388–395.

Young-Suk Lee, Bing Zhao, and Xiaoqian Luo. 2010. Constituent reordering and syntax models for English-to-Japanese statistical machine translation. In *Proceedings of COLING 2010*, pages 626–634.

Uri Lerner and Slav Petrov. 2013. Source-side classifier preordering for machine translation. In *Proceedings of EMNLP 2013*, pages 513–523.

Chi-Ho Li, Minghui Li, Dongdong Zhang, Mu Li, Ming Zhou, and Yi Guan. 2007. A probabilistic approach to syntax-based reordering for statistical machine translation. In *Proceedings of ACL 2007*, pages 720–727.

Junhui Li, Guodong Zhou, and Hwee Tou Ng. 2010. Joint syntactic and semantic parsing of Chinese. In *Proceedings of ACL 2010*, pages 1108–1117.

Junhui Li, Philip Resnik, and Hal Daumé III. 2013. Modeling syntactic and semantic structures in hierarchical phrase-based translation. In *Proceedings of HLT-NAACL 2013*, pages 540–549.

Ding Liu and Daniel Gildea. 2010. Semantic role features for machine translation. In *Proceedings of COLING 2010*, pages 716–724.

Yuval Marton and Philip Resnik. 2008. Soft syntactic constraints for hierarchical phrased-based translation. In *Proceedings of ACL-HLT 2008*, pages 1003–1011.

Markos Mylonakis and Khalil Sima'an. 2011. Learning hierarchical translation structure with linguistic annotations. In *Proceedings of ACL 2011*, pages 642–652.

ThuyLinh Nguyen and Stephan Vogel. 2013. Integrating phrase-based reordering features into a chart-based decoder for machine translation. In *Proceedings of ACL 2013*, pages 1587–1596.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of ACL 2000*, pages 440–447.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of ACL 2002*, pages 311–318.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of HLT-NAACL 2007*, pages 404–411.

Hendra Setiawan, Min Yen Kan, Haizhou Li, and Philip Resnik. 2009. Topological ordering of function words in hierarchical phrase-based translation. In *Proceedings of ACL-IJCNLP 2009*, pages 324–332.

Hendra Setiawan, Bowen Zhou, Bing Xiang, and Libin Shen. 2013. Two-neighbor orientation model with cross-boundary global contexts. In *Proceedings of ACL 2013*, pages 1264–1274.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter for sighan bakeoff 2005. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 168–171.

Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. 2009. Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In *Proceedings of ACL-IJCNLP 2009*, pages 477–485.

Karthik Visweswariah, Jiri Navratil, Jeffrey Sorensen, Vijil Chenthamarakshan, and Nandakishore Kambhatla. 2010. Syntax based reordering with automatically derived rules for improved statistical machine translation. In *Proceedings of COLING 2010*, pages 1119–1127.

Chao Wang, Michael Collins, and Philipp Koehn. 2007. Chinese syntactic reordering for statistical machine translation. In *Proceedings of EMNLP 2007*, pages 737–745.

Dekai Wu and Pascale Fung. 2009. Semantic roles for smt: A hybrid two-pass model. In *Proceedings of HLT-NAACL 2009: short papers*, pages 13–16.

Xianchao Wu, Katsuhito Sudoh, Kevin Duh, Hajime Tsukada, and Masaaki Nagata. 2011. Extracting pre-ordering rules from predicate-argument structures. In *Proceedings of IJCNLP 2011*, pages 29–37.

Fei Xia and Michael McCord. 2004. Improving a statistical mt system with automatically learned rewrite patterns. In *Proceedings of COLING 2004*, pages 508–514.

Deyi Xiong, Min Zhang, and Haizhou Li. 2012. Modeling the translation of predicate-argument structure for smt. In *Proceedings of ACL 2012*, pages 902–911.

Peng Xu, Jaeho Kang, Michael Ringgaard, and Franz Och. 2009. Using a dependency parser to improve smt for subject-object-verb languages. In *Proceedings of HLT-NAACL 2009*, pages 245–253.

Nianwen Xue and Martha Palmer. 2009. Adding semantic roles to the Chinese Treebank. *Natural Language Engineering*, 15(1):143–172.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238.

Nan Yang, Mu Li, Dongdong Zhang, and Nenghai Yu. 2012. A ranking-based approach to word reordering for statistical machine translation. In *Proceedings of ACL 2012*, pages 912–920.

Feifei Zhai, Jiajun Zhang, Yu Zhou, and Chengqing Zong. 2013. Handling ambiguities of bilingual predicate-argument structures for statistical machine translation. In *Proceedings of ACL 2013*, pages 1127–1136.

Are Two Heads Better than One? Crowdsourced Translation via a Two-Step Collaboration of Non-Professional Translators and Editors

Rui Yan, Mingkun Gao, Ellie Pavlick, and Chris Callison-Burch
Computer and Information Science Department,
University of Pennsylvania, Philadelphia, PA 19104, U.S.A.
{ruiyan,gmingkun,epavlick}@seas.upenn.edu, ccb@cis.upenn.edu

Abstract

Crowdsourcing is a viable mechanism for creating training data for machine translation. It provides a low cost, fast turnaround way of processing large volumes of data. However, when compared to professional translation, naive collection of translations from non-professionals yields low-quality results. Careful quality control is necessary for crowdsourcing to work well. In this paper, we examine the challenges of a two-step collaboration process with translation and post-editing by non-professionals. We develop graph-based ranking models that automatically select the best output from multiple redundant versions of translations and edits, and improves translation quality closer to professionals.

1 Introduction

Statistical machine translation (SMT) systems are trained using bilingual sentence-aligned parallel corpora. Theoretically, SMT can be applied to any language pair, but in practice it produces the state-of-art results only for language pairs with ample training data, like English-Arabic, English-Chinese, French-English, etc. SMT gets stuck in a severe bottleneck for many minority or 'low resource' languages with insufficient data. This drastically limits which languages SMT can be successfully applied to. Because of this, collecting parallel corpora for minor languages has become an interesting research challenge. There are various options for creating training data for new language pairs. Past approaches have examined harvesting translated documents from the web (Resnik and Smith, 2003; Uszkoreit et al., 2010; Smith et al., 2013), or discovering parallel fragments from comparable corpora (Munteanu and Marcu, 2005; Abdul-Rauf and Schwenk, 2009; Smith et al., 2010). Until relatively recently, little consideration has been given to creating parallel data from scratch. This is because the cost of hiring professional translators is prohibitively high. For instance, Germann (2001) hoped to hire professional translators to create a modest sized 100,000 word Tamil-English parallel corpus, but were stymied by the costs and the difficulty of finding good translators for a short-term commitment.

Recently, crowdsourcing has opened the possibility of translating large amounts of text at low cost using non-professional translators. Facebook localized its web site into different languages using volunteers (TechCrunch, 2008). DuoLingo turns translation into an educational game, and translates web content using its language learners (von Ahn, 2013).

Rather than relying on volunteers or gamification, NLP research into crowdsourcing translation has focused on hiring workers on the Amazon Mechanical Turk (MTurk) platform (Callison-Burch, 2009). This setup presents unique challenges, since it typically involves non-professional translators whose language skills are varied, and since it sometimes involves participants who try to cheat to get the small financial reward (Zaidan and Callison-Burch, 2011). A natural approach for trying to shore up the skills of weak bilinguals is to pair them with a native speaker of the target language to edit their translations. We review relevant research from NLP and human-computer interaction (HCI) on collaborative translation processes in Section 2. To sort good translations from bad, researchers often solicit multiple, redundant translations and then build models to try to predict which translations are the best, or which translators tend to produce the highest quality translations.

The contributions of this paper are:

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1134–1144,
Baltimore, Maryland, USA, June 23-25 2014.

- An analysis of the difficulties posed by a two-step collaboration between editors and translators in Mechanical Turk-style crowdsourcing environments. Editors vary in quality, and poor editing can be difficult to detect.

- A new graph-based algorithm for selecting the best translation among multiple translations of the same input. This method takes into account the collaborative relationship between the translators and the editors.

2 Related work

In the HCI community, several researchers have proposed protocols for collaborative translation efforts (Morita and Ishida, 2009b; Morita and Ishida, 2009a; Hu, 2009; Hu et al., 2010). These have focused on an iterative collaboration between monolingual speakers of the two languages, facilitated with a machine translation system. These studies are similar to ours in that they rely on native speakers' understanding of the target language to correct the disfluencies in poor translations. In our setup the poor translations are produced by bilingual individuals who are weak in the target language, and in their experiments the translations are the output of a machine translation system.[1]

Another significant difference is that the HCI studies assume cooperative participants. For instance, Hu et al. (2010) recruited volunteers from the International Children's Digital Library (Hourcade et al., 2003) who were all well intentioned and participated out a sense of altruism and to build a good reputation among the other volunteer translators at `childrenslibrary.org`. Our setup uses anonymous crowd workers hired on Mechanical Turk, whose motivation to participate is financial. Bernstein et al. (2010) characterized the problems with hiring editors via MTurk for a word processing application. Workers were either lazy (meaning they made only minimal edits) or overly zealous (meaning they made many unnecessary edits). Bernstein et al. (2010) addressed this problem with a three step find-fix-verify process. In the first step, workers click on one word or phrase that needed to be corrected. In the next step, a separate group of workers proposed corrections to problematic regions that had been identified by multiple workers in the first pass. In the final step, other workers would validate whether the proposed corrections were good.

Most NLP research into crowdsourcing has focused on Mechanical Turk, following pioneering work by Snow et al. (2008) who showed that the platform was a viable way of collecting data for a wide variety of NLP tasks at low cost and in large volumes. They further showed that non-expert annotations are similar to expert annotations when many non-expert labelings for the same input are aggregated, through simple voting or through weighting votes based on how closely non-experts matched experts on a small amount of calibration data. MTurk has subsequently been widely adopted by the NLP community and used for an extensive range of speech and language applications (Callison-Burch and Dredze, 2010).

Although hiring professional translators to create bilingual training data for machine translation systems has been deemed infeasible, Mechanical Turk has provided a low cost way of creating large volumes of translations (Callison-Burch, 2009; Ambati and Vogel, 2010). For instance, Zbib et al. (2012; Zbib et al. (2013) translated 1.5 million words of Levine Arabic and Egyptian Arabic, and showed that a statistical translation system trained on the dialect data outperformed a system trained on 100 times more MSA data. Post et al. (2012) used MTurk to create parallel corpora for six Indian languages for less than $0.01 per word. MTurk workers translated more than half a million words worth of Malayalam in less than a week. Several researchers have examined the use of active learning to further reduce the cost of translation (Ambati et al., 2010; Ambati, 2012; Bloodgood and Callison-Burch, 2010). Crowdsourcing allowed real studies to be conducted whereas most past active learning were simulated. Pavlick et al. (2014) conducted a large-scale demographic study of the languages spoken by workers on MTurk by translating 10,000 words in each of 100 languages. Chen and Dolan (2012) examined the steps necessary to build a persistent multilingual workforce on MTurk.

This paper is most closely related to previous work by Zaidan and Callison-Burch (2011), who showed that non-professional translators could approach the level of professional translators. They solicited multiple redundant translations from dif-

[1] A variety of HCI and NLP studies have confirmed the efficacy of monolingual or bilingual individuals post-editing of machine translation output (Callison-Burch, 2005; Koehn, 2010; Green et al., 2013). Past NLP work has also examined automatic post-editing(Knight and Chander, 1994).

Urdu translator:
According to the territory's people the pamphlets from the Taaliban had been read in the announcements in all the mosques of the Northern Wazeerastan.
English post-editor:
According to locals, the pamphlet released by the Taliban was read out on the loudspeakers of all the mosques in North Waziristan.
LDC professional:
According to the local people, the Taliban's pamphlet was read over the loudspeakers of all mosques in North Waziristan.

Table 1: Different versions of translations.

ferent Turkers for a collection of Urdu sentences that had been previously professionally translated by the Linguistics Data Consortium. They built a model to try to predict on a sentence-by-sentence and Turker-by-Turker which was the best translation or translator. They also hired US-based Turkers to edit the translations, since the translators were largely based in Pakistan and exhibited errors that are characteristic of speakers of English as a language. Zaidan and Callison-Burch (2011) observed only modest improvements when incorporating these edited translation into their model. We attempt to analyze why this is, and we proposed a new model to try to better leverage their data.

3 Crowdsourcing Translation

Setup We conduct our experiments using the data collected by Zaidan and Callison-Burch (2011). This data set consists 1,792 Urdu sentences from a variety of news and online sources, each paired with English translations provided by non-professional translators on Mechanical Turk.

Each Urdu sentence was translated redundantly by 3 distinct translators, and each translation was edited by 3 separate (native English-speaking) editors to correct for grammatical and stylistic errors. In total, this gives us 12 non-professional English candidate sentences (3 unedited, 9 edited) per original Urdu sentence. 52 different Turkers took part in the translation task, each translating 138 sentences on average. In the editing task, 320 Turkers participated, averaging 56 sentences each. For comparison, the data also includes 4 different reference translations for each source sentence, produced by professional translators.

Table 1 gives an example of an unedited translation, an edited translation, and a professional translation for the same sentence. The translations provided by translators on MTurk are generally done conscientiously, preserving the meaning of the source sentence, but typically contain simple mistakes like misspellings, typos, and awkward word choice. English-speaking editors, despite having no knowledge of the source language, are able to fix these errors. In this work, we show that the collaboration design of two heads– non-professional Urdu translators and non-professional English editors– yields better translated output than would either one working in isolation, and can better approximate the quality of professional translators.

Analysis We know from inspection that translations seem to improve with editing (Table 1). Given the data from MTurk, we explore whether this is the case in general: Do all translations improve with editing? To what extent does the individual translator and the individual editor effect the quality of the final sentence?

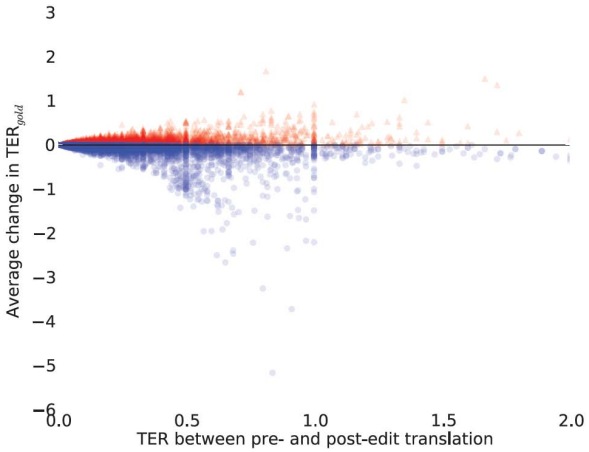


Figure 1: Relationship between editor aggressiveness and effectiveness. Each point represents an editor/translation pair. Aggressiveness (x-axis) is measured as the TER between the pre-edit and post-edit version of the translation, and effectiveness (y-axis) is measured as the average amount by which the editing reduces the translation's TER$_{gold}$. While many editors make only a few changes, those who make many changes can bring the translation substantially closer to professional quality.

We use *translation edit rate* (TER) as a measure of translation similarity. TER represents the amount of change necessary to transform one sentence into another, so a low TER means the two

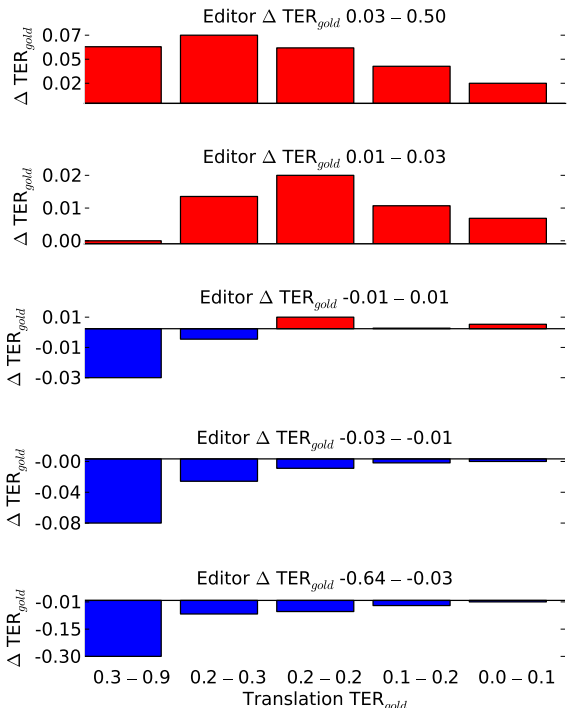


Figure 2: Effect of editing on translations of varying quality. Rows reflect bins of editors, with the worse editors (those whose changes result in increased TER$_{gold}$) on the top and the most effective editors (those whose changes result in the largest reduction in TER$_{gold}$) on the bottom. Bars reflect bins of translations, with the highest TER$_{gold}$ translations on the left, and the lowest on the right. We can see from the consistently negative Δ TER$_{gold}$ in the bottom row that good editors are able to improve both good and bad translations.

sentences are very similar. To capture the quality ("professionalness") of a translation, we take the average TER of the translation against each of our gold translations. That is, we define TER$_{gold}$ of translation t as

$$TER_{gold} = \frac{1}{4}\sum_{i=1}^{4} TER(gold_i, t) \qquad (1)$$

where a lower TER$_{gold}$ is indicative of a higher quality (more professional-sounding) translation.

We first look at editors along two dimensions: their aggressiveness and their effectiveness. Some editors may be very aggressive (they make many changes to the original translation) but still be ineffective (they fail to bring the quality of the translation closer to that of a professional). We measure aggressiveness by looking at the TER between the pre- and post-edited versions of each editor's translations; higher TER implies more aggressive editing. To measure effectiveness, we look at the change in TER$_{gold}$ that results from the editing; negative ΔTER$_{gold}$ means the editor effectively improved the quality of the translation, while positive ΔTER$_{gold}$ means the editing actually brought the translation further from our gold standard.

Figure 1 shows the relationship between these two qualities for individual editor/translation pairs. We see that while most translations require only a few edits, there are a large number of translations which improve substantially after heavy editing. This trend conforms to our intuition that editing is most useful when the translation has much room for improvement, and opens the question of whether good editors can offer improvements to translations of all qualities.

To address this question, we split our translations into 5 bins, based on their TER$_{gold}$. We also split our editors into 5 bins, based on their effectiveness (i.e. the average amount by which their editing reduces TER$_{gold}$). Figure 2 shows the degree to which editors at each level are able to improve the translations from each bin. We see that good editors are able to make improvements to translations of all qualities, but that good editing has the greatest impact on lower quality translations. This result suggests that finding good editor/translator pairs, rather than good editors and good translators in isolation, should produce the best translations overall. Figure 3 gives an example of how an initially medium-quality translation, when combined with good editing, produces a better result than the higher-quality translation paired with mediocre editing.

4 Problem Formulation

The problem definition of the crowdsourcing translation task is straightforward: given a set of candidate translations for a source sentence, we want to choose the best output translation.

This output translation is the result of the combined translation and editing stages. Therefore, our method operates over a heterogeneous network that includes translators and post-editors as well as the translated sentences that they produce. We frame the problem as follows. We form two graphs: the first graph (G_T) represents Turkers (translator/editor pairs) as nodes; the second graph (G_C) represents candidate translated and

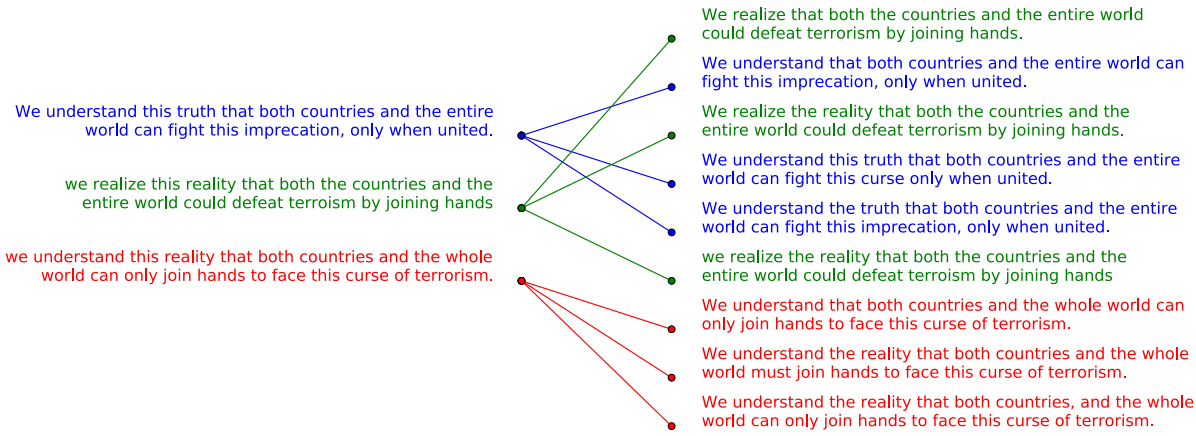


Figure 3: Three alternative translations (left) and the edited versions of each (right). Each edit on the right was produced by a different editor. Order reflects the TER_{gold} of each translation, with the lowest TER_{gold} on the top. Some translators receive low TER_{gold} scores due to superficial errors, which can be easily improved through editing. In the above example, the middle-ranked translation (green) becomes the best translation after being revised by a good editor.

post-edited sentences (henceforth "candidates") as nodes. These two graphs, G_T and G_C are combined as subgraphs of a third graph (G_{TC}). Edges in G_{TC} connect author pairs (nodes in G_T) to the candidate that they produced (nodes in G_C). Together, G_T, G_C, and G_{TC} define a co-ranking problem (Yan et al., 2012a; Yan et al., 2011b; Yan et al., 2012b) with linkage establishment (Yan et al., 2011a; Yan et al., 2012c), which we define formally as follows.

Let G denote the heterogeneous graph with nodes V and edges E. Let $G = (V,E) = (V_T, V_C, E_T, E_C, E_{TC})$. G is divided into three subgraphs, G_T, G_C, and G_{TC}. $G_C = (V_C, E_C)$ is a weighted undirected graph representing the candidates and their lexical relationships to one another. Let V_C denote a collection of translated and edited candidates, and E_C the lexical similarity between the candidates (see Section 4.3 for details). $G_T = (V_T, E_T)$ is a weighted undirected graph representing collaborations between Turkers. V_T is the set of translator/editor pairs. Edges E_T connect translator/editor pairs in V_T which share a translator and/or editor. Each collaboration (i.e. each node in V_T) produces a candidate (i.e. a node in V_C). $G_{TC} = (V_{TC}, E_{TC})$ is an unweighted bipartite graph that ties G_T and G_C together and represents "authorship". The graph G consists of nodes $V_{TC} = V_T \cup V_C$ and edges E_{TC} connecting each candidate with its authoring translator/post-editor pair. The three sub-networks (G_T, G_C, and G_{TC}) are illustrated in Figure 4.

4.1 Inter-Graph Ranking

The framework includes three random walks, one on G_T, one on G_C and one on G_{TC}. A random walk on a graph is a Markov chain, its states being the vertices of the graph. It can be described by a stochastic square matrix, where the dimension is the number of vertices in the graph, and the entries describe the transition probabilities from one vertex to the next. The mutual reinforcement framework couples the two random walks on G_T and G_C that rank candidates and Turkers in isolation. The ranking method allows us to obtain a global ranking by taking into account the intra-/inter-component dependencies. In the following sections, we describe how we obtain the rankings on G_T and G_C, and then move on to discuss how the two are coupled.

Our algorithm aims to capture the following intuitions. A candidate is important if 1) it is similar to many of the other proposed candidates and 2) it is authored by better qualified translators and/or post-editors. Analogously, a translator/editor pair is believed to be better qualified if 1) the editor is collaborating with a good translator and vice versa and 2) the pair has authored important candidates. This ranking schema is actually a reinforced process across the heterogeneous graphs. We use two vectors $\mathbf{c} = [\pi(c)]_{|c|\times 1}$ and $\mathbf{t} = [\pi(t)]_{|t|\times 1}$ to denote the saliency scores $\pi(.)$ of candidates and Turker pairs. The above-mentioned intuitions can be formulated as follows:

- **Homogeneity.** We use adjacency matrix

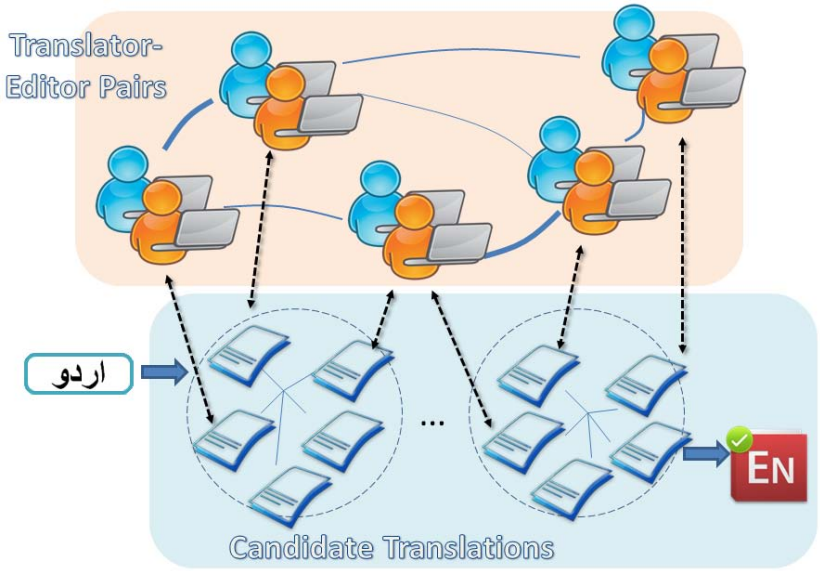


Figure 4: 2-step collaborative crowdsourcing translation model based on graph ranking framework including three sub-networks. The undirected links between users denotes *translation-editing* collaboration. The undirected links between candidate translations indicate lexical similarity between candidates. A bipartite graph ties candidate and Turker networks together by authorship (to make the figure clearer, some linkage is omitted). A dashed circle indicates the group of candidate translations for a single source sentence to translate.

$[M]_{|\mathbf{c}|\times|\mathbf{c}|}$ to describe the homogeneous affinity between candidates and $[N]_{|\mathbf{t}|\times|\mathbf{t}|}$ to describe the affinity between Turkers.

$$\mathbf{c} \propto M^T\mathbf{c}, \qquad \mathbf{t} \propto N^T\mathbf{t} \tag{2}$$

where $c = |V_C|$ is the number of vertices in the candidate graph and $t = |V_T|$ is the number of vertices in the Turker graph. The adjacency matrix $[M]$ denotes the transition probabilities between candidates, and analogously matrix $[N]$ denotes the affinity between Turker collaboration pairs.

• **Heterogeneity.** We use an adjacency matrix $[\hat{W}]_{|\mathbf{c}|\times|\mathbf{t}|}$ and $[\bar{W}]_{|\mathbf{t}|\times|\mathbf{c}|}$ to describe the authorship between the output candidate and the producer Turker pair from both of the candidate-to-Turker and Turker-to-candidate perspectives.

$$\mathbf{c} \propto \hat{W}^T\mathbf{t}, \qquad \mathbf{t} \propto \bar{W}^T\mathbf{c} \tag{3}$$

All affinity matrices will be defined in the next section. By fusing the above equations, we can have the following iterative calculation in matrix forms. For numerical computation of the saliency scores, the initial scores of all sentences and Turkers are set to 1 and the following two steps are alternated until convergence to select the best candidate.

Step 1: compute the saliency scores of candidates, and then normalize using ℓ-1 norm.

$$\begin{aligned}\mathbf{c}^{(n)} &= (1-\lambda)M^T\mathbf{c}^{(n-1)} + \lambda\hat{W}\mathbf{t}^{(n-1)} \\ \mathbf{c}^{(n)} &= \mathbf{c}^{(n)}/||\mathbf{c}^{(n)}||_1\end{aligned} \tag{4}$$

Step 2: compute the saliency scores of Turker pairs, and then normalize using ℓ-1 norm.

$$\begin{aligned}\mathbf{t}^{(n)} &= (1-\lambda)N^T\mathbf{t}^{(n-1)} + \lambda\bar{W}\mathbf{c}^{(n-1)} \\ \mathbf{t}^{(n)} &= \mathbf{t}^{(n)}/||\mathbf{t}^{(n)}||_1\end{aligned} \tag{5}$$

where λ specifies the relative contributions to the saliency score trade-off between the homogeneous affinity and the heterogeneous affinity. In order to guarantee the convergence of the iterative form, we must force the transition matrix to be stochastic and irreducible. To this end, we must make the $\mathbf{c}$ and $\mathbf{t}$ *column stochastic* (Langville and Meyer, 2004). $\mathbf{c}$ and $\mathbf{t}$ are therefore normalized after each iteration of Equation (4) and (5).

4.2 Intra-Graph Ranking

The standard PageRank algorithm starts from an arbitrary node and randomly selects to either follow a random out-going edge (considering the weighted transition matrix) or to jump to a random node (treating all nodes with equal probability).

In a simple random walk, it is assumed that all nodes in the transitional matrix are equi-probable before the walk starts. Then **c** and **t** are calculated as:

$$\mathbf{c} = \mu M^T \mathbf{c} + (1-\mu)\frac{\mathbf{1}}{|V_C|} \quad (6)$$

and

$$\mathbf{t} = \mu N^T \mathbf{t} + (1-\mu)\frac{\mathbf{1}}{|V_T|} \quad (7)$$

where **1** is a vector with all elements equaling to 1 and the size is correspondent to the size of V_C or V_T. μ is the damping factor usually set to 0.85, as in the PageRank algorithm.

4.3 Affinity Matrix Establishment

We introduce the affinity matrix calculation, including homogeneous affinity (i.e., M, N) and heterogeneous affinity (i.e., $\hat{W}, \bar{W}$).

As discussed, we model the collection of candidates as a weighted undirected graph, G_C, in which nodes in the graph represent candidate sentences and edges represent lexical relatedness. We define an edge's weight to be the cosine similarity between the candidates represented by the nodes that it connects. The adjacency matrix M describes such a graph, with each entry corresponding to the weight of an edge.

$$\begin{aligned} \mathcal{F}(c_i, c_j) &= \frac{c_i \cdot c_j}{||c_i||||c_j||} \\ M_{ij} &= \frac{\mathcal{F}(c_i, c_j)}{\sum_k \mathcal{F}(c_i, c_k)} \end{aligned} \quad (8)$$

where $\mathcal{F}(.)$ is the cosine similarity and c is a term vector corresponding to a candidate. We treat a candidate as a short document and weight each term with *tf.idf* (Manning et al., 2008), where *tf* is the term frequency and *idf* is the inverse document frequency.

The Turker graph, G_T, is an undirected graph whose edges represent "collaboration." Formally, let t_i and t_j be two translator/editor pairs; we say that pair t_i "collaborates with" pair t_j (and therefore, there is an edge between t_i and t_j) if t_i and t_j share either a translator or an editor (or share both a translator and an editor). Let the function $\mathcal{I}(t_i, t_j)$ denote the number of "collaborations" ($\#col$) between t_i and t_j.

$$\mathcal{I}(t_i, t_j) = \begin{cases} \#col & (e_{ij} \in E_T) \\ 0 & \text{otherwise} \end{cases}, \quad (9)$$

Then the adjacency matrix N is then defined as

$$N_{ij} = \frac{\mathcal{I}(t_i, t_j)}{\sum_k \mathcal{I}(t_i, t_k)} \quad (10)$$

In the bipartite candidate-Turker graph G_{TC}, the entry $E_{TC}(i, j)$ is an indicator function denoting whether the candidate c_i is generated by t_j:

$$\mathcal{A}(c_i, t_j) = \begin{cases} 1 & (e_{ij} \in E_{TC}) \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

Through E_{TC} we define the weight matrices $\bar{W}_{ij}$ and $\hat{W}_{ij}$, containing the conditional probabilities of transitions from c_i to t_j and vice versa:

$$\begin{aligned} \bar{W}_{ij} &= \frac{\mathcal{A}(c_i, t_j)}{\sum_k \mathcal{A}(c_i, t_k)}, \\ \hat{W}_{ij} &= \frac{\mathcal{A}(c_i, t_j)}{\sum_k \mathcal{A}(c_k, t_j)} \end{aligned} \quad (12)$$

5 Evaluation

We are interested in testing our random walk method, which incorporates information from both the candidate translations and from the Turkers. We want to test two versions of our proposed collaborative co-ranking method: 1) based on the unedited translations only and 2) based on the edited sentences after translator/editor collaborations.

Metric Since we have four professional translation sets, we can calculate the Bilingual Evaluation Understudy (BLEU) score (Papineni et al., 2002) for one professional translator (P1) using the other three (P2,3,4) as a reference set. We repeat the process four times, scoring each professional translator against the others, to calculate the expected range of professional quality translation. In the following sections, we evaluate each of our methods by calculating BLEU scores against the same four sets of three reference translations. Therefore, each number reported in our experimental results is an average of four numbers, corresponding to the four possible ways of choosing 3 of the 4 reference sets. This allows us to compare the BLEU score achieved by our methods against the BLEU scores achievable by professional translators.

Baselines As a naive baseline, we choose one candidate translation at random for each input Urdu sentence. To establish an upper bound for our methods, and to determine if there exist high-quality Turker translations at all, we compute four

Reference (Avg.)	42.51
Oracle (Seg-Trans)	44.93
Oracle (Seg-Trans+Edit)	48.44
Oracle (Turker-Trans)	38.66
Oracle (Turker-Trans+Edit)	39.16
Random	30.52
Lowest TER	35.78
Graph Ranking (Trans)	38.88
Graph Ranking (Trans+Edit)	**41.43**

Table 2: Overall BLEU performance for all methods (with and without post-editing). The highlighted result indicates the best performance, which is based on both candidate sentences and Turker information.

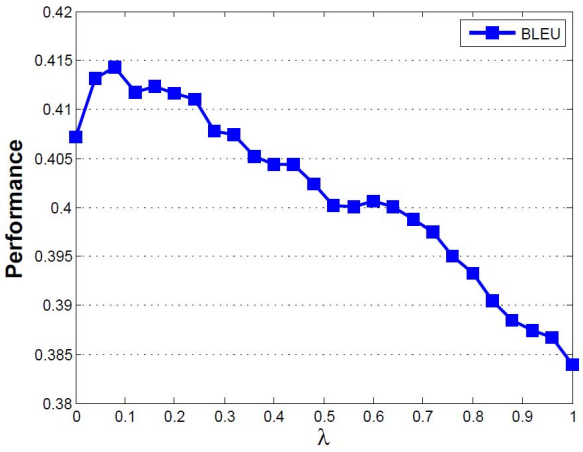


Figure 5: Effect of candidate-Turker coupling (λ) on BLEU score.

oracle scores. The first oracle operates at the segment level on the sentences produced by translators only: for each source segment, we choose from the translations the one that scores highest (in terms of BLEU) against the reference sentences. The second oracle is applied similarly, but chooses from the candidates produced by the collaboration of translator/post-editor pairs. The third oracle operates at the worker level: for each source segment, we choose from the translations the one provided by the worker whose translations (over all sentences) score the highest on average. The fourth oracle also operates at the worker level, but selects from sentences produced by translator/post-editor collaborations. These oracle methods represent ideal solutions under our scenario. We also examine two voting-inspired methods. The first method selects the translation with the minimum average TER (Snover et al., 2006) against the other translations; intuitively, this would represent the "consensus" translation. The second method selects the translation generated by the Turker who, on average, provides translations with the minimum average TER.

Results A summary of our results in given in Table 2. As expected, random selection yields bad performance, with a BLEU score of 30.52. The oracles indicate that there is usually an acceptable translation from the Turkers for any given sentence. Since the oracles select from a small group of only 4 translations per source segment, they are not overly optimistic, and rather reflect the true potential of the collected translations. On average, the reference translations give a score of 42.38. To put this in perspective, the output of a state-of-the-art machine translation system (the syntax-based variant of Joshua) achieves a score of 26.91, which is reported in (Zaidan and Callison-Burch, 2011). The approach which selects the translations with the minimum average TER (Snover et al., 2006) against the other three translations (the "consensus" translation) achieves BLEU scores of 35.78.

Using the raw translations without post-editing, our graph-based ranking method achieves a BLEU score of 38.89, compared to Zaidan and Callison-Burch (2011)' s reported score of 28.13, which they achieved using a linear feature-based classification. Their linear classifier achieved a reported score of 39.06[2] when combining information from both translators and editors. In contrast, our proposed graph-based ranking framework achieves a score of 41.43 when using the same information. This boost in BLEU score confirms our intuition that the hidden collaboration networks between candidate translations and transltor/editor pairs are indeed useful.

Parameter Tuning There are two parameters in our experimental setups: μ controls the probability of starting a new random walk and λ controls the coupling between the candidate and Turker subgraphs. We set the damping factor μ to 0.85, following the standard PageRank paradigm. In order to determine a value for λ, we used the average BLEU, computed against the professional refer-

[2]Note that the data we used in our experiments are slightly different, by discarding nearly 100 NULL sentences in the raw data. We do not re-implement this baseline but report the results from the paper directly. According to our experiments, most of the results generated by baselines and oracles are very close to the previously reported values.

Plain ranking	38.89
w/o collaboration	38.88
Shared translator	41.38
Shared post-editor	41.29
Shared Turker	**41.43**

Table 3: Variations of all component settings.

ence translations, as a tuning metric. We experimented with values of λ ranging from 0 to 1, with a step size of 0.05 (Figure 5). Small λ values place little emphasis on the candidate/Turker coupling, whereas larger values rely more heavily on the co-ranking. Overall, we observed better performance with values within the range of 0.05-0.15. This suggests that both sources of information– the candidate itself and its authors– are important for the crowdsourcing translation task. In all of our reported results, we used the $\lambda = 0.1$.

Analysis We examine the relative contribution of each component of our approach on the overall performance. We first examine the centroid-based ranking on the candidate sub-graph (G_C) alone to see the effect of voting among translated sentences; we denote this strategy as *plain ranking*. Then we incorporate the standard random walk on the Turker graph (G_T) to include the structural information but without yet including any collaboration information; that is, we incorporate information from G_T and G_C without including edges linking the two together. The co-ranking paradigm is exactly the same as the framework described in Section 3.2, but with simplified structures.

Finally, we examine the two-step collaboration based candidate-Turker graph using several variations on edge establishment. As before, the nodes are the translator/post-editor working pairs. We investigate three settings in which 1) edges connect two nodes when they share only a translator, 2) edges connect two nodes when they share only a post-editor, and 3) edges connect two nodes when they share either a translator or a post-editor. These results are summarized in Table 3.

Interestingly, we observe that when modeling the linkage between the collaboration pairs, connecting Turker pairs which share either a translator or the post-editor achieves better performance than connecting pairs that share only translators or connecting pairs which share only editors. This result supports the intuition that a denser collaboration matrix will help propagate saliency to good translators/post-editors and hence provides better predictions for candidate quality.

6 Conclusion

We have proposed an algorithm for using a two-step collaboration between non-professional translators and post-editors to obtain professional-quality translations. Our method, based on a co-ranking model, selects the best crowdsourced translation from a set of candidates, and is capable of selecting translations which near professional quality.

Crowdsourcing can play a pivotal role in future efforts to create parallel translation datasets. In addition to its benefits of cost and scalability, crowdsourcing provides access to languages that currently fall outside the scope of statistical machine translation research. In future work on crowdsourced translation, further benefits in quality improvement and cost reduction could stem from 1) building ground truth data sets based on high-quality Turkers' translations and 2) identifying when sufficient data has been collected for a given input, to avoid soliciting unnecessary redundant translations.

Acknowledgements

This material is based on research sponsored by a DARPA Computer Science Study Panel phase 3 award entitled "Crowdsourcing Translation" (contract D12PC00368). The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements by DARPA or the U.S. Government. This research was supported by the Johns Hopkins University Human Language Technology Center of Excellence and through gifts from Microsoft, Google and Facebook.

References

Sadaf Abdul-Rauf and Holger Schwenk. 2009. On the use of comparable corpora to improve SMT performance. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 16–23, March.

Vamshi Ambati and Stephan Vogel. 2010. Can crowds build parallel corpora for machine translation systems? In *Workshop on Creating Speech and Language Data with MTurk*.

Vamshi Ambati, Stephan Vogel, and Jaime G Carbonell. 2010. Active learning and crowd-sourcing for machine translation. In *LREC*, volume 11, pages 2169–2174. Citeseer.

Vamshi Ambati. 2012. *Active Learning and Crowdsourcing for Machine Translation in Low Resource Scenarios*. Ph.D. thesis, Language Technologies Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Michael S. Bernstein, Greg Little, Robert C. Miller, Bjrn Hartmann, Mark S. Ackerman, David R. Karger, David Crowell, and Katrina Panovich. 2010. Soylent: a word processor with a crowd inside. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*.

Michael Bloodgood and Chris Callison-Burch. 2010. Large-scale cost-focused active learning for statistical machine translation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.

Chris Callison-Burch and Mark Dredze. 2010. Creating speech and language data with Amazon's Mechanical Turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 1–12, June.

Chris Callison-Burch. 2005. Linear B system description for the 2005 NIST MT evaluation exercise. In *Proceedings of Machine Translation Evaluation Workshop*.

Chris Callison-Burch. 2009. Fast, cheap, and creative: Evaluating translation quality using amazon's mechanical turk. In *Proceedings of EMNLP*.

David L. Chen and William B. Dolan. 2012. Building a persistent workforce on mechanical turk for multilingual data collection. In *Proceedings of the Human Computer Interaction International Conference*.

Ulrich Germann. 2001. Building a statistical machine translation system from scratch: How much bang for the buck can we expect? In *Proceedings of the Workshop on Data-driven Methods in Machine Translation - Volume 14*, DMMT '01, pages 1–8.

Spence Green, Jeffrey Heer, and Christopher D. Manning. 2013. The efficacy of human post-editing for language translation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 439–448.

Juan Pablo Hourcade, Benjamin B Bederson, Allison Druin, Anne Rose, Allison Farber, and Yoshifumi Takayama. 2003. The international children's digital library: viewing digital books online. *Interacting with Computers*, 15(2):151–167.

Chang Hu, Benjamin B. Bederson, and Philip Resnik. 2010. Translation by iterative collaboration between monolingual users. In *Proceedings of ACM SIGKDD Workshop on Human Computation (HCOMP)*.

Chang Hu, Benjamin B. Bederson, Philip Resnik, and Yakov Kronrod. 2011. Monotrans2: A new human computation system to support monolingual translation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 1133–1136.

Chang Hu. 2009. Collaborative translation by monolingual users. In *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '09, pages 3105–3108.

Martin Kay. 1998. The proper place of men and machines in language translation. *Machine Translation*, 12(1/2):3–23, January.

Kevin Knight and Ishwar Chander. 1994. Automated postediting of documents. In *In Proceedings of AAAI*.

Philipp Koehn. 2010. Enabling monolingual translators: Post-editing vs. options. In *HLT-NAACL'10*, pages 537–545, June.

Amy N Langville and Carl D Meyer. 2004. Deeper inside pagerank. *Internet Mathematics*, 1(3):335–380.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Ann Irvine, Sanjeev Khudanpur, Lane Schwartz, Wren Thornton, Ziyuan Wang, Jonathan Weese, and Omar Zaidan. 2010. Joshua 2.0: A toolkit for parsing-based machine translation with syntax, semirings, discriminative training and other goodies. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 133–137, July.

Annie Louis and Ani Nenkova. 2013. What makes writing great? first experiments on article quality prediction in the science journalism domain. *Transactions of Association for Computational Linguistics*.

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge.

Daisuke Morita and Toru Ishida. 2009a. Collaborative translation by monolinguals with machine translators. In *Proceedings of the 14th International Conference on Intelligent User Interfaces*, IUI '09, pages 361–366.

Daisuke Morita and Toru Ishida. 2009b. Designing protocols for collaborative translation. In *International Conference on Principles of Practice in Multi-Agent Systems (PRIMA-09)*, pages 17–32. Springer.

Dragos Stefan Munteanu and Daniel Marcu. 2005. Improving machine translation performance by exploiting non-parallel corpora. *Comput. Linguist.*, 31(4):477–504, December.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318.

Ellie Pavlick, Matt Post, Ann Irvine, Dmitry Kachaev, and Chris Callison-Burch. 2014. The language demographics of Amazon Mechanical Turk. *Transactions of the Association for Computational Linguistics (TACL)*, 2(Feb):79–92.

Matt Post, Chris Callison-Burch, and Miles Osborne. 2012. Constructing parallel corpora for six Indian languages via crowdsourcing. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*.

Philip Resnik and Noah A. Smith. 2003. The web as a parallel corpus. *Computational Linguistics*, 29(3):349–380, September.

Jason R. Smith, Chris Quirk, and Kristina Toutanova. 2010. Extracting parallel sentences from comparable corpora using document level alignment. In *HLT-NAACL'10*, pages 403–411.

Jason Smith, Herve Saint-Amand, Magdalena Plamada, Philipp Koehn, Chris Callison-Burch, and Adam Lopez. 2013. Dirt cheap web-scale parallel text from the Common Crawl. In *Proceedings of the 2013 Conference of the Association for Computational Linguistics (ACL 2013)*, July.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, pages 223–231.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast—but is it good?: Evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 254–263.

TechCrunch. 2008. Facebook taps users to create translated versions of site, January.

Jakob Uszkoreit, Jay M. Ponte, Ashok C. Popat, and Moshe Dubiner. 2010. Large scale parallel document mining for machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 1101–1109.

Luis von Ahn. 2013. Duolingo: Learn a language for free while helping to translate the web. In *Proceedings of the 2013 International Conference on Intelligent User Interfaces*, IUI '13, pages 1–2.

Rui Yan, Liang Kong, Congrui Huang, Xiaojun Wan, Xiaoming Li, and Yan Zhang. 2011a. Timeline generation through evolutionary trans-temporal summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 433–443.

Rui Yan, Xiaojun Wan, Jahna Otterbacher, Liang Kong, Xiaoming Li, and Yan Zhang. 2011b. Evolutionary timeline summarization: A balanced optimization framework via iterative substitution. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 745–754.

Rui Yan, Mirella Lapata, and Xiaoming Li. 2012a. Tweet recommendation with graph co-ranking. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 516–525.

Rui Yan, Xiaojun Wan, Mirella Lapata, Wayne Xin Zhao, Pu-Jen Cheng, and Xiaoming Li. 2012b. Visualizing timelines: Evolutionary summarization via iterative reinforcement between text and image streams. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 275–284.

Rui Yan, Zi Yuan, Xiaojun Wan, Yan Zhang, and Xiaoming Li. 2012c. Hierarchical graph summarization: leveraging hybrid information through visible and invisible linkage. In *PAKDD'12*, pages 97–108. Springer.

Omar F. Zaidan and Chris Callison-Burch. 2011. Crowdsourcing translation: Professional quality from non-professionals. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1220–1229.

Rabih Zbib, Erika Malchiodi, Jacob Devlin, David Stallard, Spyros Matsoukas, Richard Schwartz, John Makhoul, Omar F. Zaidan, and Chris Callison-Burch. 2012. Machine translation of Arabic dialects. In *The 2012 Conference of the North American Chapter of the Association for Computational Linguistics*.

Rabih Zbib, Gretchen Markiewicz, Spyros Matsoukas, Richard Schwartz, and John Makhoul. 2013. Systematic comparison of professional and crowdsourced reference translations for machine translation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Atlanta, Georgia.

Xin Wayne Zhao, Yanwei Guo, Rui Yan, Yulan He, and Xiaoming Li. 2013. Timeline generation with social attention. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 1061–1064.

A Generalized Language Model as the Combination of Skipped n-grams and Modified Kneser-Ney Smoothing

Rene Pickhardt, Thomas Gottron, Martin Körner, Steffen Staab
Institute for Web Science and Technologies,
University of Koblenz-Landau, Germany
{rpickhardt,gottron,mkoerner,staab}@uni-koblenz.de

Paul Georg Wagner and Till Speicher
Typology GbR
mail@typology.de

Abstract

We introduce a novel approach for building language models based on a systematic, recursive exploration of skip n-gram models which are interpolated using modified Kneser-Ney smoothing. Our approach generalizes language models as it contains the classical interpolation with lower order models as a special case. In this paper we motivate, formalize and present our approach. In an extensive empirical experiment over English text corpora we demonstrate that our generalized language models lead to a substantial reduction of perplexity between 3.1% and 12.7% in comparison to traditional language models using modified Kneser-Ney smoothing. Furthermore, we investigate the behaviour over three other languages and a domain specific corpus where we observed consistent improvements. Finally, we also show that the strength of our approach lies in its ability to cope in particular with sparse training data. Using a very small training data set of only 736 KB text we yield improvements of even 25.7% reduction of perplexity.

1 Introduction motivation

Language Models are a probabilistic approach for predicting the occurrence of a sequence of words. They are used in many applications, e.g. word prediction (Bickel et al., 2005), speech recognition (Rabiner and Juang, 1993), machine translation (Brown et al., 1990), or spelling correction (Mays et al., 1991). The task language models attempt to solve is the estimation of a probability of a given sequence of words $w_1^l = w_1, \ldots, w_l$. The probability $P(w_1^l)$ of this sequence can be broken down into a product of conditional probabilities:

$$\begin{aligned} P(w_1^l) =& P(w_1) \cdot P(w_2|w_1) \cdot \ldots \cdot P(w_l|w_1 \cdots w_{l-1}) \\ =& \prod_{i=1}^{l} P(w_i|w_1 \cdots w_{i-1}) \end{aligned} \quad (1)$$

Because of combinatorial explosion and data sparsity, it is very difficult to reliably estimate the probabilities that are conditioned on a longer subsequence. Therefore, by making a Markov assumption the true probability of a word sequence is only approximated by restricting conditional probabilities to depend only on a local context w_{i-n+1}^{i-1} of $n-1$ preceding words rather than the full sequence w_1^{i-1}. The challenge in the construction of language models is to provide reliable estimators for the conditional probabilities. While the estimators can be learnt—using, e.g., a maximum likelihood estimator over n-grams obtained from training data—the obtained values are not very reliable for events which may have been observed only a few times or not at all in the training data.

Smoothing is a standard technique to overcome this data sparsity problem. Various smoothing approaches have been developed and applied in the context of language models. Chen and Goodman (Chen and Goodman, 1999) introduced modified Kneser-Ney Smoothing, which up to now has been considered the state-of-the-art method for language modelling over the last 15 years. Modified Kneser-Ney Smoothing is an interpolating method which combines the estimated conditional probabilities $P(w_i|w_{i-n+1}^{i-1})$ recursively with lower order models involving a shorter local context w_{i-n+2}^{i-1} and their estimate for $P(w_i|w_{i-n+2}^{i-1})$. The motivation for using lower order models is that shorter contexts may be observed more often and, thus, suffer less from data sparsity. However, a single rare word towards the end of the local context will always cause the context to be observed rarely in the training data and hence will lead to an unreliable estimation.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1145–1154,
Baltimore, Maryland, USA, June 23-25 2014.

Because of Zipfian word distributions, most words occur very rarely and hence their true probability of occurrence may be estimated only very poorly. One word that appears at the end of a local context w_{i-n+1}^{i-1} and for which only a poor approximation exists may adversely affect the conditional probabilities in language models of all lengths — leading to severe errors even for smoothed language models. Thus, the idea motivating our approach is to involve several lower order models which systematically leave out one position in the context (one may think of replacing the affected word in the context with a wildcard) instead of shortening the sequence only by one word at the beginning.

This concept of introducing gaps in n-grams is referred to as skip n-grams (Ney et al., 1994; Huang et al., 1993). Among other techniques, skip n-grams have also been considered as an approach to overcome problems of data sparsity (Goodman, 2001). However, to best of our knowledge, language models making use of skip n-grams models have never been investigated to their full extent and over different levels of lower order models. Our approach differs as we consider all possible combinations of gaps in a local context and interpolate the higher order model with all possible lower order models derived from adding gaps in all different ways.

In this paper we make the following contributions:

1. We provide a framework for using modified Kneser-Ney smoothing in combination with a systematic exploration of lower order models based on skip n-grams.

2. We show how our novel approach can indeed easily be interpreted as a generalized version of the current state-of-the-art language models.

3. We present a large scale empirical analysis of our generalized language models on eight data sets spanning four different languages, namely, a wikipedia-based text corpus and the JRC-Acquis corpus of legislative texts.

4. We empirically observe that introducing skip n-gram models may reduce perplexity by 12.7% compared to the current state-of-the-art using modified Kneser-Ney models on large data sets. Using small training data sets we observe even higher reductions of perplexity of up to 25.6%.

The rest of the paper is organized as follows. We start with reviewing related work in Section 2. We will then introduce our generalized language models in Section 3. After explaining the evaluation methodology and introducing the data sets in Section 4 we will present the results of our evaluation in Section 5. In Section 6 we discuss why a generalized language model performs better than a standard language model. Finally, in Section 7 we summarize our findings and conclude with an overview of further interesting research challenges in the field of generalized language models.

2 Related Work

Work related to our generalized language model approach can be divided in two categories: various smoothing techniques for language models and approaches making use of skip n-grams.

Smoothing techniques for language models have a long history. Their aim is to overcome data sparsity and provide more reliable estimators—in particular for rare events. The Good Turing estimator (Good, 1953), deleted interpolation (Jelinek and Mercer, 1980), Katz backoff (Katz, 1987) and Kneser-Ney smoothing (Kneser and Ney, 1995) are just some of the approaches to be mentioned. Common strategies of these approaches are to either backoff to lower order models when a higher order model lacks sufficient training data for good estimation, to interpolate between higher and lower order models or to interpolate with a prior distribution. Furthermore, the estimation of the amount of unseen events from rare events aims to find the right weights for interpolation as well as for discounting probability mass from unreliable estimators and to retain it for unseen events.

The state of the art is a modified version of Kneser-Ney smoothing introduced in (Chen and Goodman, 1999). The modified version implements a recursive interpolation with lower order models, making use of different discount values for more or less frequently observed events. This variation has been compared to other smoothing techniques on various corpora and has shown to outperform competing approaches. We will review modified Kneser-Ney smoothing in Section 2.1 in more detail as we reuse some ideas to define our generalized language model.

Smoothing techniques which do not rely on using lower order models involve clustering (Brown et al., 1992; Ney et al., 1994), i.e. grouping together similar words to form classes of words, as well as skip n-grams (Ney et al., 1994; Huang et al., 1993). Yet other approaches make use of permutations of the word order in n-grams (Schukat-Talamazzini et al., 1995; Goodman, 2001).

Skip n-grams are typically used to incorporate long distance relations between words. Introducing the possibility of gaps between the words in an n-gram allows for capturing word relations beyond the level of n consecutive words without an exponential increase in the parameter space. However, with their restriction on a subsequence of words, skip n-grams are also used as a technique to overcome data sparsity (Goodman, 2001). In related work different terminology and different definitions have been used to describe skip n-grams. Variations modify the number of words which can be skipped between elements in an n-gram as well as the manner in which the skipped words are determined (e.g. fixed patterns (Goodman, 2001) or functional words (Gao and Suzuki, 2005)).

The impact of various extensions and smoothing techniques for language models is investigated in (Goodman, 2001; Goodman, 2000). In particular, the authors compared Kneser-Ney smoothing, Katz backoff smoothing, caching, clustering, inclusion of higher order n-grams, sentence mixture and skip n-grams. They also evaluated combinations of techniques, for instance, using skip n-gram models in combination with Kneser-Ney smoothing. The experiments in this case followed two paths: (1) interpolating a 5-gram model with lower order distribution introducing a single gap and (2) interpolating higher order models with skip n-grams which retained only combinations of two words. Goodman reported on small data sets and in the best case a moderate improvement of cross entropy in the range of 0.02 to 0.04.

In (Guthrie et al., 2006), the authors investigated the increase of observed word combinations when including skips in n-grams. The conclusion was that using skip n-grams is often more effective for increasing the number of observations than increasing the corpus size. This observation aligns well with our experiments.

2.1 Review of Modified Kneser-Ney Smoothing

We briefly recall modified Kneser-Ney Smoothing as presented in (Chen and Goodman, 1999). Modified Kneser-Ney implements smoothing by interpolating between higher and lower order n-gram language models. The highest order distribution is interpolated with lower order distribution as follows:

$$P_{\text{MKN}}(w_i|w_{i-n+1}^{i-1}) = \frac{\max\{c(w_{i-n+1}^{i}) - D(c(w_{i-n+1}^{i})), 0\}}{c(w_{i-n+1}^{i-1})} + \gamma_{high}(w_{i-n+1}^{i-1})\hat{P}_{\text{MKN}}(w_i|w_{i-n+2}^{i-1}) \quad (2)$$

where $c(w_{i-n+1}^{i})$ provides the frequency count that sequence w_{i-n+1}^{i} occurs in training data, D is a discount value (which depends on the frequency of the sequence) and γ_{high} depends on D and is the interpolation factor to mix in the lower order distribution[1]. Essentially, interpolation with a lower order model corresponds to leaving out the first word in the considered sequence. The lower order models are computed differently using the notion of continuation counts rather than absolute counts:

$$\hat{P}_{\text{MKN}}(w_i|(w_{i-n+1}^{i-1})) = \frac{\max\{N_{1+}(\bullet w_{i-n+1}^{i}) - D(c(w_{i-n+1}^{i})), 0\}}{N_{1+}(\bullet w_{i-n+1}^{i-1}\bullet)} + \gamma_{mid}(w_{i-n+1}^{i-1})\hat{P}_{\text{MKN}}(w_i|w_{i-n+2}^{i-1})) \quad (3)$$

where the continuation counts are defined as $N_{1+}(\bullet w_{i-n+1}^{i}) = |\{w_{i-n} : c(w_{i-n}^{i}) > 0\}|$, i.e. the number of different words which precede the sequence w_{i-n+1}^{i}. The term γ_{mid} is again an interpolation factor which depends on the discounted probability mass D in the first term of the formula.

3 Generalized Language Models

3.1 Notation for Skip n-gram with k Skips

We express skip n-grams using an operator notation. The operator ∂_i applied to an n-gram removes the word at the i-th position. For instance: $\partial_3 w_1 w_2 w_3 w_4 = w_1 w_2 \_ w_4$, where $\_$ is used as wildcard placeholder to indicate a removed word. The wildcard operator allows for

[1] The factors γ and D are quite technical and lengthy. As they do not play a significant role for understanding our novel approach we refer to Appendix A for details.

larger number of matches. For instance, when $c(w_1 w_2 w_{3a} w_4) = x$ and $c(w_1 w_2 w_{3b} w_4) = y$ then $c(w_1 w_2 \_ w4) \geq x + y$ since at least the two sequences $w_1 w_2 w_{3a} w_4$ and $w_1 w_2 w_{3b} w_4$ match the sequence $w_1 w_2 \_ w_4$. In order to align with standard language models the skip operator applied to the first word of a sequence will remove the word instead of introducing a wildcard. In particular the equation $\partial_1 w_{i-n+1}^i = w_{i-n+2}^i$ holds where the right hand side is the subsequence of w_{i-n+1}^i omitting the first word. We can thus formulate the interpolation step of modified Kneser-Ney smoothing using our notation as $\hat{P}_{\text{MKN}}(w_i|w_{i-n+2}^{i-1}) = \hat{P}_{\text{MKN}}(w_i|\partial_1 w_{i-n+1}^{i-1})$.

Thus, our skip n-grams correspond to n-grams of which we only use k words, after having applied the skip operators $\partial_{i_1} \dots \partial_{i_{n-k}}$

3.2 Generalized Language Model

Interpolation with lower order models is motivated by the problem of data sparsity in higher order models. However, lower order models omit only the first word in the local context, which might not necessarily be the cause for the overall n-gram to be rare. This is the motivation for our generalized language models to not only interpolate with one lower order model, where the first word in a sequence is omitted, but also with all other skip n-gram models, where one word is left out. Combining this idea with modified Kneser-Ney smoothing leads to a formula similar to (2).

$$P_{\text{GLM}}(w_i|w_{i-n+1}^{i-1}) = \frac{\max\{c(w_{i-n+1}^i) - D(c(w_{i-n+1}^i)), 0\}}{c(w_{i-n+1}^{i-1})} + \gamma_{high}(w_{i-n+1}^{i-1}) \sum_{j=1}^{n-1} \frac{1}{n-1} \hat{P}_{\text{GLM}}(w_i|\partial_j w_{i-n+1}^{i-1}) \quad (4)$$

The difference between formula (2) and formula (4) is the way in which lower order models are interpolated.

Note, the sum over all possible positions in the context w_{i-n+1}^{i-1} for which we can skip a word and the according lower order models $P_{\text{GLM}}(w_i|\partial_j(w_{i-n+1}^{i-1}))$. We give all lower order models the same weight $\frac{1}{n-1}$.

The same principle is recursively applied in the lower order models in which some words of the full n-gram are already skipped. As in modified Kneser-Ney smoothing we use continuation counts for the lower order models, incorporating the skip operator also for these counts. Incorporating this directly into modified Kneser-Ney smoothing leads in the second highest model to:

$$\hat{P}_{\text{GLM}}(w_i|\partial_j(w_{i-n+1}^{i-1})) = \frac{\max\{N_{1+}(\partial_j(w_{i-n}^i)) - D(c(\partial_j(w_{i-n+1}^i))), 0\}}{N_{1+}(\partial_j(w_{i-n+1}^{i-1})\bullet)} + \gamma_{mid}(\partial_j(w_{i-n+1}^{i-1})) \sum_{\substack{k=1 \\ k \neq j}}^{n-1} \frac{1}{n-2} \hat{P}_{\text{GLM}}(w_i|\partial_j \partial_k(w_{i-n+1}^{i-1})) \quad (5)$$

Given that we skip words at different positions, we have to extend the notion of the count function and the continuation counts. The count function applied to a skip n-gram is given by $c(\partial_j(w_{i-n}^i)) = \sum_{w_j} c(w_{i-n}^i)$, i.e. we aggregate the count information over all words which fill the gap in the n-gram. Regarding the continuation counts we define:

$$N_{1+}(\partial_j(w_{i-n}^i)) = |\{w_{i-n+j-1} : c(w_{i-n}^i) > 0\}| \quad (6)$$

$$N_{1+}(\partial_j(w_{i-n}^{i-1})\bullet) = |\{(w_{i-n+j-1}, w_i) : c(w_{i-n}^i) > 0\}| \quad (7)$$

As lowest order model we use—just as done for traditional modified Kneser-Ney (Chen and Goodman, 1999)—a unigram model interpolated with a uniform distribution for unseen words.

The overall process is depicted in Figure 1, illustrating how the higher level models are recursively smoothed with several lower order ones.

4 Experimental Setup and Data Sets

To evaluate the quality of our generalized language models we empirically compare their ability to explain sequences of words. To this end we use text corpora, split them into test and training data, build language models as well as generalized language models over the training data and apply them on the test data. We employ established metrics, such as cross entropy and perplexity. In the following we explain the details of our experimental setup.

4.1 Data Sets

For evaluation purposes we employed eight different data sets. The data sets cover different domains and languages. As languages we considered English (*en*), German (*de*), French (*fr*), and Italian (*it*). As general domain data set we used the full collection of articles from Wikipedia (*wiki*) in the corresponding languages. The download dates of the dumps are displayed in Table 1.

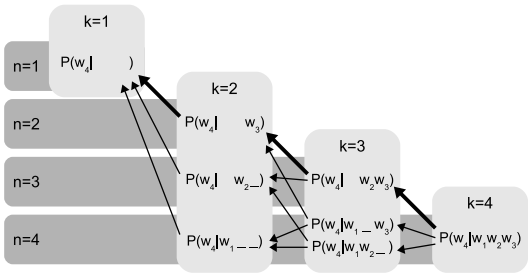


Figure 1: Interpolation of models of different order and using skip patterns. The value of n indicates the length of the raw n-grams necessary for computing the model, the value of k indicates the number of words actually used in the model. The wild card symbol _ marks skipped words in an n-gram. The arrows indicate how a higher order model is interpolated with lower order models which skips one word. The bold arrows correspond to interpolation of models in traditional modified Kneser-Ney smoothing. The lighter arrows illustrate the additional interpolations introduced by our generalized language models.

de	en	fr	it
Nov 22nd	Nov 04th	Nov 20th	Nov 25th

Table 1: Download dates of Wikipedia snapshots in November 2013.

Special purpose domain data are provided by the multi-lingual JRC-Acquis corpus of legislative texts (*JRC*) (Steinberger et al., 2006). Table 2 gives an overview of the data sets and provides some simple statistics of the covered languages and the size of the collections.

Corpus	Statistics	
	total words in Mio.	unique words in Mio.
wiki-de	579	9.82
JRC-de	30.9	0.66
wiki-en	1689	11.7
JRC-en	39.2	0.46
wiki-fr	339	4.06
JRC-fr	35.8	0.46
wiki-it	193	3.09
JRC-it	34.4	0.47

Table 2: Word statistics and size of of evaluation corpora

The data sets come in the form of structured text corpora which we cleaned from markup and tokenized to generate word sequences. We filtered the word tokens by removing all character sequences which did not contain any letter, digit or common punctuation marks. Eventually, the word token sequences were split into word sequences of length n which provided the basis for the training and test sets for all algorithms. Note that we did not perform case-folding nor did we apply stemming algorithms to normalize the word forms. Also, we did our evaluation using case sensitive training and test data. Additionally, we kept all tokens for named entities such as names of persons or places.

4.2 Evaluation Methodology

All data sets have been randomly split into a training and a test set on a sentence level. The training sets consist of 80% of the sentences, which have been used to derive n-grams, skip n-grams and corresponding continuation counts for values of n between 1 and 5. Note that we have trained a prediction model for each data set individually. From the remaining 20% of the sequences we have randomly sampled a separate set of $100,000$ sequences of 5 words each. These test sequences have also been shortened to sequences of length 3, and 4 and provide a basis to conduct our final experiments to evaluate the performance of the different algorithms.

We learnt the generalized language models on the same split of the training corpus as the standard language model using modified Kneser-Ney smoothing and we also used the same set of test sequences for a direct comparison. To ensure rigour and openness of research the data set for training as well as the test sequences and the entire source code is open source. [2] [3] [4] We compared the probabilities of our language model implementation (which is a subset of the generalized language model) using KN as well as MKN smoothing with the Kyoto Language Model Toolkit [5]. Since we got the same results for small n and small data sets we believe that our implementation is correct.

In a second experiment we have investigated the impact of the size of the training data set. The wikipedia corpus consists of 1.7 bn. words.

[2] http://west.uni-koblenz.de/Research

[3] https://github.com/renepickhardt/generalized-language-modeling-toolkit

[4] http://glm.rene-pickhardt.de

[5] http://www.phontron.com/kylm/

Thus, the 80% split for training consists of 1.3 bn. words. We have iteratively created smaller training sets by decreasing the split factor by an order of magnitude. So we created 8% / 92% and 0.8% / 99.2% split, and so on. We have stopped at the $0.008\%/99.992\%$ split as the training data set in this case consisted of less words than our 100k test sequences which we still randomly sampled from the test data of each split. Then we trained a generalized language model as well as a standard language model with modified Kneser-Ney smoothing on each of these samples of the training data. Again we have evaluated these language models on the same random sample of $100,000$ sequences as mentioned above.

4.3 Evaluation Metrics

As evaluation metric we use *perplexity*: a standard measure in the field of language models (Manning and Schütze, 1999). First we calculate the *cross entropy* of a trained language model given a test set using

$$H(P_{\texttt{alg}}) = -\sum_{s \in T} P_{\texttt{MLE}}(s) \cdot \log_2 P_{\texttt{alg}}(s) \quad (8)$$

Where $P_{\texttt{alg}}$ will be replaced by the probability estimates provided by our generalized language models and the estimates of a language model using modified Kneser-Ney smoothing. $P_{\texttt{MLE}}$, instead, is a maximum likelihood estimator of the test sequence to occur in the test corpus. Finally, T is the set of test sequences. The perplexity is defined as:

$$\textit{Perplexity}(P_{\texttt{alg}}) = 2^{H(P_{\texttt{alg}})} \quad (9)$$

Lower perplexity values indicate better results.

5 Results

5.1 Baseline

As a baseline for our generalized language model (GLM) we have trained standard language models using modified Kneser-Ney Smoothing (MKN). These models have been trained for model lengths 3 to 5. For unigram and bigram models MKN and GLM are identical.

5.2 Evaluation Experiments

The perplexity values for all data sets and various model orders can be seen in Table 3. In this table we also present the relative reduction of perplexity in comparison to the baseline.

	model length		
Experiments	$n=3$	$n=4$	$n=5$
wiki-de MKN	1074.1	778.5	597.1
wiki-de GLM	**1031.1**	**709.4**	**521.5**
rel. change	4.0%	8.9%	12.7%
JRC-de MKN	235.4	138.4	94.7
JRC-de GLM	**229.4**	**131.8**	**86.0**
rel. change	2.5%	4.8%	9.2%
wiki-en MKN	586.9	404	307.3
wiki-en GLM	**571.6**	**378.1**	**275**
rel. change	2.6%	6.1%	10.5%
JRC-en MKN	147.2	82.9	54.6
JRC-en GLM	**145.3**	**80.6**	**52.5**
rel. change	1.3%	2.8%	3.9%
wiki-fr MKN	538.6	385.9	298.9
wiki-fr GLM	**526.7**	**363.8**	**272.9**
rel. change	2.2%	5.7%	8.7%
JRC-fr MKN	155.2	92.5	63.9
JRC-fr GLM	**153.5**	**90.1**	**61.7**
rel. change	1.1%	2.5%	3.5%
wiki-it MKN	738.4	532.9	416.7
wiki-it GLM	**718.2**	**500.7**	**382.2**
rel. change	2.7%	6.0%	8.3%
JRC-it MKN	177.5	104.4	71.8
JRC-it GLM	**175.1**	**101.8**	**69.6**
rel. change	1.3%	2.6%	3.1%

Table 3: Absolute perplexity values and relative reduction of perplexity from MKN to GLM on all data sets for models of order 3 to 5

As we can see, the GLM clearly outperforms the baseline for all model lengths and data sets. In general we see a larger improvement in performance for models of higher orders ($n = 5$). The gain for 3-gram models, instead, is negligible. For German texts the increase in performance is the highest (12.7%) for a model of order 5. We also note that GLMs seem to work better on broad domain text rather than special purpose text as the reduction on the wiki corpora is constantly higher than the reduction of perplexity on the JRC corpora.

We made consistent observations in our second experiment where we iteratively shrank the size of the training data set. We calculated the relative reduction in perplexity from MKN to GLM

for various model lengths and the different sizes of the training data. The results for the English Wikipedia data set are illustrated in Figure 2.

We see that the GLM performs particularly well on small training data. As the size of the training data set becomes smaller (even smaller than the evaluation data), the GLM achieves a reduction of perplexity of up to 25.7% compared to language models with modified Kneser-Ney smoothing on the same data set. The absolute perplexity values for this experiment are presented in Table 4.

	model length		
Experiments	$n=3$	$n=4$	$n=5$
80% MKN	586.9	404	307.3
80% GLM	**571.6**	**378.1**	**275**
rel. change	2.6%	6.5%	10.5%
8% MKN	712.6	539.8	436.5
8% GLM	**683.7**	**492.8**	**382.5**
rel. change	4.1%	8.7%	12.4%
0.8% MKN	894.0	730.0	614.1
0.8% GLM	**838.7**	**650.1**	**528.7**
rel. change	6.2%	10.9%	13.9%
0.08% MKN	1099.5	963.8	845.2
0.08% GLM	**996.6**	**820.7**	**693.4**
rel. change	9.4%	14.9%	18.0%
0.008% MKN	1212.1	1120.5	1009.6
0.008% GLM	**1025.6**	**875.5**	**750.3**
rel. change	15.4%	21.9%	25.7%

Table 4: Absolute perplexity values and relative reduction of perplexity from MKN to GLM on shrunk training data sets for the English Wikipedia for models of order 3 to 5

Our theory as well as the results so far suggest that the GLM performs particularly well on sparse training data. This conjecture has been investigated in a last experiment. For each model length we have split the test data of the largest English Wikipedia corpus into two disjoint evaluation data sets. The data set *unseen* consists of all test sequences which have never been observed in the training data. The set *observed* consists only of test sequences which have been observed at least once in the training data. Again we have calculated the perplexity of each set. For reference, also the values of the complete test data set are shown in Table 5.

	model length		
Experiments	$n=3$	$n=4$	$n=5$
$\text{MKN}^{\text{complete}}$	586.9	404	307.3
$\text{GLM}^{\text{complete}}$	**571.6**	**378.1**	**275**
rel. change	2.6%	6.5%	10.5%
$\text{MKN}^{\text{unseen}}$	14696.8	2199.8	846.1
$\text{GLM}^{\text{unseen}}$	**13058.7**	**1902.4**	**714.4**
rel. change	11.2%	13.5%	15.6%
$\text{MKN}^{\text{observed}}$	**220.2**	**88.0**	**43.4**
$\text{GLM}^{\text{observed}}$	220.6	88.3	43.5
rel. change	-0.16%	-0.28%	-0.15%

Table 5: Absolute perplexity values and relative reduction of perplexity from MKN to GLM for the complete and split test file into observed and unseen sequences for models of order 3 to 5. The data set is the largest English Wikipedia corpus.

As expected we see the overall perplexity values rise for the *unseen* test case and decline for the *observed* test case. More interestingly we see that the relative reduction of perplexity of the GLM over MKN increases from 10.5% to 15.6% on the *unseen* test case. This indicates that the superior performance of the GLM on small training corpora and for higher order models indeed comes from its good performance properties with regard to sparse training data. It also confirms that our motivation to produce lower order n-grams by omitting not only the first word of the local context but systematically all words has been fruitful. However, we also see that for the *observed* sequences the GLM performs slightly worse than MKN. For the observed cases we find the relative change to be negligible.

6 Discussion

In our experiments we have observed an improvement of our generalized language models over classical language models using Kneser-Ney smoothing. The improvements have been observed for different languages, different domains as well as different sizes of the training data. In the experiments we have also seen that the GLM performs well in particular for small training data sets and sparse data, encouraging our initial motivation. This feature of the GLM is of particular value, as data sparsity becomes a more and more immanent problem for higher values of n. This known fact is underlined also by the statis-

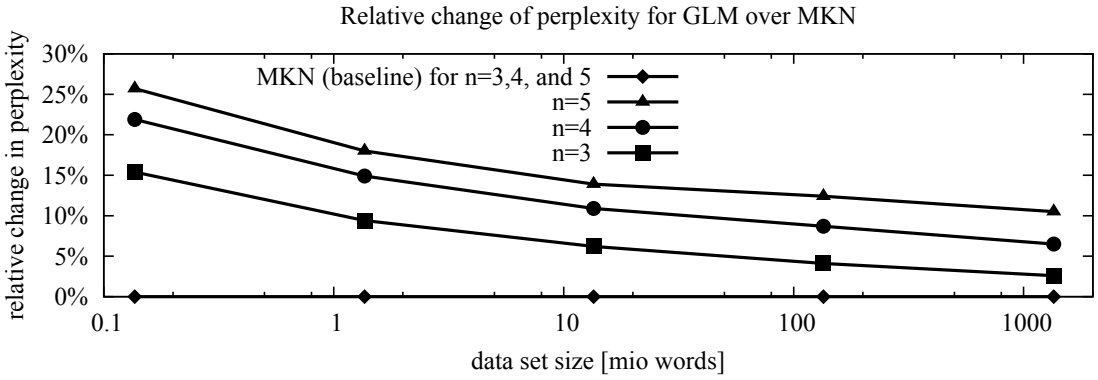


Figure 2: Variation of the size of the training data on 100k test sequences on the English Wikipedia data set with different model lengths for GLM.

tics shown in Table 6. The fraction of total n-grams which appear only once in our Wikipedia corpus increases for higher values of n. However, for the same value of n the skip n-grams are less rare. Our generalized language models leverage this additional information to obtain more reliable estimates for the probability of word sequences.

w_1^n	**total**	**unique**
w_1	0.5%	64.0%
$w_1 w_2$	5.1%	68.2%
$w_1 \_ w_3$	8.0%	79.9%
$w_1 \_\_ w_4$	9.6%	72.1%
$w_1 \_\_\_ w_5$	10.1%	72.7%
$w_1 w_2 w_3$	21.1%	77.5%
$w_1 \_ w_3 w_4$	28.2%	80.4%
$w_1 w_2 \_ w_4$	28.2%	80.7%
$w_1 \_\_ w_4 w_5$	31.7%	81.9%
$w_1 \_ w_3 \_ w_5$	35.3%	83.0%
$w_1 w_2 \_\_ w_5$	31.5%	82.2%
$w_1 w_2 w_3 w_4$	44.7%	85.4%
$w_1 \_ w_3 w_4 w_5$	52.7%	87.6%
$w_1 w_2 \_ w_4 w_5$	52.6%	88.0%
$w_1 w_2 w_3 \_ w_5$	52.3%	87.7%
$w_1 w_2 w_3 w_4 w_5$	64.4%	90.7%

Table 6: Percentage of generalized n-grams which occur only once in the English Wikipedia corpus. Total means a percentage relative to the total amount of sequences. Unique means a percentage relative to the amount of unique sequences of this pattern in the data set.

Beyond the general improvements there is an additional path for benefitting from generalized language models. As it is possible to better leverage the information in smaller and sparse data sets, we can build smaller models of competitive performance. For instance, when looking at Table 4 we observe the 3-gram MKN approach on the full training data set to achieve a perplexity of 586.9. This model has been trained on 7 GB of text and the resulting model has a size of 15 GB and 742 Mio. entries for the count and continuation count values. Looking for a GLM with comparable but better performance we see that the 5-gram model trained on 1% of the training data has a perplexity of 528.7. This GLM model has a size of 9.5 GB and contains only 427 Mio. entries. So, using a far smaller set of training data we can build a smaller model which still demonstrates a competitive performance.

7 Conclusion and Future Work

7.1 Conclusion

We have introduced a novel generalized language model as the systematic combination of skip n-grams and modified Kneser-Ney smoothing. The main strength of our approach is the combination of a simple and elegant idea with an an empirically convincing result. Mathematically one can see that the GLM includes the standard language model with modified Kneser-Ney smoothing as a sub model and is consequently a real generalization.

In an empirical evaluation, we have demonstrated that for higher orders the GLM outperforms MKN for all test cases. The relative improvement in perplexity is up to 12.7% for large data sets. GLMs also performs particularly well on small and sparse sets of training data. On a very

small training data set we observed a reduction of perplexity by 25.7%. Our experiments underline that the generalized language models overcome in particular the weaknesses of modified Kneser-Ney smoothing on sparse training data.

7.2 Future work

A desirable extension of our current definition of GLMs will be the combination of different lower lower order models in our generalized language model using different weights for each model. Such weights can be used to model the statistical reliability of the different lower order models. The value of the weights would have to be chosen according to the probability or counts of the respective skip n-grams.

Another important step that has not been considered yet is compressing and indexing of generalized language models to improve the performance of the computation and be able to store them in main memory. Regarding the scalability of the approach to very large data sets we intend to apply the Map Reduce techniques from (Heafield et al., 2013) to our generalized language models in order to have a more scalable calculation.

This will open the path also to another interesting experiment. Goodman (Goodman, 2001) observed that increasing the length of n-grams in combination with modified Kneser-Ney smoothing did not lead to improvements for values of n beyond 7. We believe that our generalized language models could still benefit from such an increase. They suffer less from the sparsity of long n-grams and can overcome this sparsity when interpolating with the lower order skip n-grams while benefiting from the larger context.

Finally, it would be interesting to see how applications of language models—like next word prediction, machine translation, speech recognition, text classification, spelling correction, e.g.—benefit from the better performance of generalized language models.

Acknowledgements

We would like to thank Heinrich Hartmann for a fruitful discussion regarding notation of the skip operator for n-grams. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013), REVEAL (Grant agree number 610928).

References

Steffen Bickel, Peter Haider, and Tobias Scheffer. 2005. Predicting sentences using n-gram language models. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 193–200, Stroudsburg, PA, USA. Association for Computational Linguistics.

Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Fredrick Jelinek, John D Lafferty, Robert L Mercer, and Paul S Roossin. 1990. A statistical approach to machine translation. *Computational linguistics*, 16(2):79–85.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–479, December.

Stanley Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical report, TR-10-98, Harvard University, August.

Stanley Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393.

Jianfeng Gao and Hisami Suzuki. 2005. Long distance dependency in language modeling: An empirical study. In Keh-Yih Su, Junichi Tsujii, Jong-Hyeok Lee, and OiYee Kwong, editors, *Natural Language Processing IJCNLP 2004*, volume 3248 of *Lecture Notes in Computer Science*, pages 396–405. Springer Berlin Heidelberg.

Irwin J. Good. 1953. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3-4):237–264.

Joshua T. Goodman. 2000. Putting it all together: language model combination. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings. 2000 IEEE International Conference on*, volume 3, pages 1647–1650 vol.3.

Joshua T. Goodman. 2001. A bit of progress in language modeling – extended version. Technical Report MSR-TR-2001-72, Microsoft Research.

David Guthrie, Ben Allison, Wei Liu, Louise Guthrie, and York Wilks. 2006. A closer look at skip-gram modelling. In *Proceedings LREC'2006*, pages 1222–1225.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified kneser-ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

Xuedong Huang, Fileno Alleva, Hsiao-Wuen Hon, Mei-Yuh Hwang, Kai-Fu Lee, and Ronald Rosenfeld. 1993. The sphinx-ii speech recognition system: an overview. *Computer Speech & Language*, 7(2):137 – 148.

F. Jelinek and R.L. Mercer. 1980. Interpolated estimation of markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice*, pages 381–397.

S. Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 35(3):400–401.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. MIT Press, Cambridge, MA, USA.

Eric Mays, Fred J Damerau, and Robert L Mercer. 1991. Context based spelling correction. *Information Processing & Management*, 27(5):517–522.

Hermann Ney, Ute Essen, and Reinhard Kneser. 1994. On structuring probabilistic dependences in stochastic language modelling. *Computer Speech & Language*, 8(1):1 – 38.

Lawrence Rabiner and Biing-Hwang Juang. 1993. *Fundamentals of Speech Recognition*. Prentice Hall.

Ernst-Günter Schukat-Talamazzini, R Hendrych, Ralf Kompe, and Heinrich Niemann. 1995. Permugram language models. In *Fourth European Conference on Speech Communication and Technology*.

Ralf Steinberger, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Tomaz Erjavec, Dan Tufis, and Daniel Varga. 2006. The jrc-acquis: A multilingual aligned parallel corpus with 20+ languages. In *LREC'06: Proceedings of the 5th International Conference on Language Resources and Evaluation*.

A Discount Values and Weights in Modified Kneser Ney

The discount value $D(c)$ used in formula (2) is defined as (Chen and Goodman, 1999):

$$D(c) = \begin{cases} 0 & \text{if } c = 0 \\ D_1 & \text{if } c = 1 \\ D_2 & \text{if } c = 2 \\ D_{3+} & \text{if } c > 2 \end{cases} \tag{10}$$

The discounting values D_1, D_2, and D_{3+} are defined as (Chen and Goodman, 1998)

$$D_1 = 1 - 2Y\frac{n_2}{n_1} \tag{11a}$$

$$D_2 = 2 - 3Y\frac{n_3}{n_2} \tag{11b}$$

$$D_{3+} = 3 - 4Y\frac{n_4}{n_3} \tag{11c}$$

with $Y = \frac{n_1}{n_1+n_2}$ and n_i is the total number of n-grams which appear exactly i times in the training data. The weight $\gamma_{high}(w_{i-n+1}^{i-1})$ is defined as:

$$\gamma_{high}(w_{i-n+1}^{i-1}) = \frac{D_1 N_1(w_{i-n+1}^{i-1}\bullet) + D_2 N_2(w_{i-n+1}^{i-1}\bullet) + D_{3+} N_{3+}(w_{i-n+1}^{i-1}\bullet)}{c(w_{i-n+1}^{i-1})} \tag{12}$$

And the weight $\gamma_{mid}(w_{i-n+1}^{i-1})$ is defined as:

$$\gamma_{mid}(w_{i-n+1}^{i-1}) = \frac{D_1 N_1(w_{i-n+1}^{i-1}\bullet) + D_2 N_2(w_{i-n+1}^{i-1}\bullet) + D_{3+} N_{3+}(w_{i-n+1}^{i-1}\bullet)}{N_{1+}(\bullet w_{i-n+1}^{i-1}\bullet)} \tag{13}$$

where $N_1(w_{i-n+1}^{i-1}\bullet)$, $N_2(w_{i-n+1}^{i-1}\bullet)$, and $N_{3+}(w_{i-n+1}^{i-1}\bullet)$ are analogously defined to $N_{1+}(w_{i-n+1}^{i-1}\bullet)$.

A Semiparametric Gaussian Copula Regression Model for Predicting Financial Risks from Earnings Calls

William Yang Wang
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
yww@cs.cmu.edu

Zhenhao Hua
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
zhua@cs.cmu.edu

Abstract

Earnings call summarizes the financial performance of a company, and it is an important indicator of the future financial risks of the company. We quantitatively study how earnings calls are correlated with the financial risks, with a special focus on the financial crisis of 2009. In particular, we perform a text regression task: given the transcript of an earnings call, we predict the volatility of stock prices from the week after the call is made. We propose the use of *copula*: a powerful statistical framework that separately models the uniform marginals and their complex multivariate stochastic dependencies, while not requiring any prior assumptions on the distributions of the covariate and the dependent variable. By performing *probability integral transform*, our approach moves beyond the standard count-based bag-of-words models in NLP, and improves previous work on text regression by incorporating the correlation among local features in the form of semiparametric Gaussian copula. In experiments, we show that our model significantly outperforms strong linear and non-linear discriminative baselines on three datasets under various settings.

1 Introduction

Predicting the risks of publicly listed companies is of great interests not only to the traders and analysts on the Wall Street, but also virtually anyone who has investments in the market (Kogan et al., 2009). Traditionally, analysts focus on quantitative modeling of historical trading data. Today, even though earnings calls transcripts are abundantly available, their distinctive communicative practices (Camiciottoli, 2010), and correlations with the financial risks, in particular, future stock performances (Price et al., 2012), are not well studied in the past.

Earnings calls are conference calls where a listed company discusses the financial performance. Typically, a earnings call contains two parts: the senior executives first report the operational outcomes, as well as the current financial performance, and then discuss their perspectives on the future of the company. The second part of the teleconference includes a question answering session where the floor will be open to investors, analysts, and other parties for inquiries. The question we ask is that, even though each earnings call has distinct styles, as well as different speakers and mixed formats, can we use earnings calls to predict the financial risks of the company in the limited future?

Given a piece of earnings call transcript, we investigate a semiparametric approach for automatic prediction of future financial risk[1]. To do this, we formulate the problem as a text regression task, and use a Gaussian copula with probability integral transform to model the uniform marginals and their dependencies. Copula models (Schweizer and Sklar, 1983; Nelsen, 1999) are often used by statisticians (Genest and Favre, 2007; Liu et al., 2012; Masarotto and Varin, 2012) and economists (Chen and Fan, 2006) to study the bivariate and multivariate stochastic dependency among random variables, but they are very new to the machine learning (Ghahramani et al., 2012; Han et al., 2012; Xiang and Neville, 2013; Lopez-paz et al., 2013) and related communities (Eickhoff et al., 2013). To the best of our knowledge, even though the term "copula" is named for the resemblance to grammatical copulas in linguistics, copula models have not been explored in the NLP community. To evaluate the performance of our approach, we compare with a standard squared loss linear regression baseline, as well as strong baselines such as linear and non-linear support

[1] In this work, the risk is defined as the measured volatility of stock prices from the week following the earnings call teleconference. See details in Section 5.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1155–1165,
Baltimore, Maryland, USA, June 23-25 2014.

vector machines (SVMs) that are widely used in text regression tasks. By varying different experimental settings on three datasets concerning different periods of the Great Recession from 2006-2013, we empirically show that our approach significantly outperforms the baselines by a wide margin. Our main contributions are:

- We are among the first to formally study transcripts of earnings calls to predict financial risks.
- We propose a novel semiparametric Gaussian copula model for text regression.
- Our results significantly outperform standard linear regression and strong SVM baselines.
- By varying the number of dimensions of the covariates and the size of the training data, we show that the improvements over the baselines are robust across different parameter settings on three datasets.

In the next section, we outline related work in modeling financial reports and text regression. In Section 3, the details of the semiparametric copula model are introduced. We then describe the dataset and dependent variable in this study, and the experiments are shown in Section 6. We discuss the results and findings in Section 7 and then conclude in Section 8.

2 Related Work

Fung et al. (2003) are among the first to study SVM and text mining methods in the market prediction domain, where they align financial news articles with multiple time series to simulate the 33 stocks in the Hong Kong Hang Seng Index. However, text regression in the financial domain have not been explored until recently. Kogan et al. (2009) model the SEC-mandated annual reports, and performs linear SVM regression with ϵ-insensitive loss function to predict the measured volatility. Another recent study (Wang et al., 2013) uses exactly the same max-margin regression technique, but with a different focus on the financial sentiment. Using the same dataset, Tsai and Wang (2013) reformulate the regression problem as a text ranking problem. Note that all these regression studies above investigate the SEC-mandated annual reports, which are very different from the earnings calls in many aspects such as length, format, vocabulary, and genre. Most recently, Xie et al. (2013) have proposed the use of frame-level semantic features to understand financial news, but they treat the stock movement prediction problem as a binary classification task. Broadly speaking, our work is also aligned to recent studies that make use of social media data to predict the stock market (Bollen et al., 2011; Zhang et al., 2011).

Despite our financial domain, our approach is more relevant to text regression. Traditional discriminative models, such as linear regression and linear SVM, have been very popular in various text regression tasks, such as predicting movie revenues from reviews (Joshi et al., 2010), understanding the geographic lexical variation (Eisenstein et al., 2010), and predicting food prices from menus (Chahuneau et al., 2012). The advantage of these models is that the estimation of the parameters is often simple, the results are easy to interpret, and the approach often yields strong performances. While these approaches have merits, they suffer from the problem of not explicitly modeling the correlations and interactions among random variables, which in some sense, corresponding to the impractical assumption of independent and identically distributed (i.i.d) of the data. For example, when bag-of-word-unigrams are present in the feature space, it is easier if one does not explicitly model the stochastic dependencies among the words, even though doing so might hurt the predictive power, while the variance from the correlations among the random variables is not explained.

3 Copula Models for Text Regression

In NLP, many statistical machine learning methods that capture the dependencies among random variables, including topic models (Blei et al., 2003; Lafferty and Blei, 2005; Wang et al., 2012), always have to make assumptions with the underlying distributions of the random variables, and make use of informative priors. This might be rather restricting the expressiveness of the model in some sense (Reisinger et al., 2010). On the other hand, once such assumptions are removed, another problem arises — they might be prone to errors, and suffer from the overfitting issue. Therefore, coping with the tradeoff between expressiveness and overfitting, seems to be rather important in statistical approaches that capture stochastic dependency.

Our proposed semiparametric copula regression model takes a different perspective. On one hand, copula models (Nelsen, 1999) seek to explicitly model the dependency of random variables by separating the marginals and their correlations. On the other hand, it does not make use of any as-

sumptions on the distributions of the random variables, yet, the copula model is still expressive. This nice property essentially allows us to fuse distinctive lexical, syntactic, and semantic feature sets naturally into a single compact model.

From an information-theoretic point of view (Shannon, 1948), various problems in text analytics can be formulated as estimating the probability mass/density functions of tokens in text. In NLP, many of the probabilistic text models work in the discrete space (Church and Gale, 1995; Blei et al., 2003), but our model is different: since the text features are sparse, we first perform kernel density estimates to smooth out the zeroing items, and then calculate the empirical cumulative distribution function (CDF) of the random variables. By doing this, we are essentially performing *probability integral transform*— an important statistical technique that moves beyond the count-based bag-of-words feature space to marginal cumulative density functions space. Last but not least, by using a parametric copula, in our case, the Gaussian copula, we reduce the computational cost from fully nonparametric methods, and explicitly model the correlations among the covariate and the dependent variable.

In this section, we first briefly look at the theoretical foundations of copulas, including the Sklar's theorem. Then we describe the proposed semiparametric Gaussian copula text regression model. The algorithmic implementation of our approach is introduced at the end of this section.

3.1 The Theory of Copula

In the statistics literature, copula is widely known as a family of distribution function. The idea behind copula theory is that the cumulative distribution function (CDF) of a random vector can be represented in the form of uniform marginal cumulative distribution functions, and a copula that connects these marginal CDFs, which describes the correlations among the input random variables. However, in order to have a valid multivariate distribution function regardless of n-dimensional covariates, not every function can be used as a copula function. The central idea behind copula, therefore, can be summarize by the Sklar's theorem and the corollary.

Theorem 1 (Sklar's Theorem (1959)) *Let F be the joint cumulative distribution function of n random variables $X_1, X_2, ..., X_n$. Let the corresponding marginal cumulative distribution functions of the random variable be $F_1(x_1), F_2(x_2), ..., F_n(x_n)$. Then, if the marginal functions are continuous, there exists a unique copula C, such that*

$$F(x_1, ..., x_n) = C[F_1(x_1), ..., F_n(x_n)]. \quad (1)$$

Furthermore, if the distributions are continuous, the multivariate dependency structure and the marginals might be separated, and the copula can be considered independent of the marginals (Joe, 1997; Parsa and Klugman, 2011). Therefore, the copula does not have requirements on the marginal distributions, and any arbitrary marginals can be combined and their dependency structure can be modeled using the copula. The inverse of Sklar's Theorem is also true in the following:

Corollary 1 *If there exists a copula $C : (0,1)^n$ and marginal cumulative distribution functions $F_1(x_1), F_2(x_2), ..., F_n(x_n)$, then $C[F_1(x_1), ..., F_n(x_n)]$ defines a multivariate cumulative distribution function.*

3.2 Semiparametric Gaussian Copula Models

The Non-Parametric Estimation

We formulate the copula regression model as follows. Assume we have n random variables of text features $X_1, X_2, ..., X_n$. The problem is that text features are sparse, so we need to perform nonparametric kernel density estimation to smooth out the distribution of each variable. Let $f_1, f_2, ..., f_n$ be the unknown density, we are interested in deriving the shape of these functions. Assume we have m samples, the kernel density estimator can be defined as:

$$\hat{f}_h(x) = \frac{1}{m}\sum_{i=1}^{m} K_h(x - x_i) \quad (2)$$

$$= \frac{1}{mh}\sum_{i=1}^{m} K\left(\frac{x - x_i}{h}\right) \quad (3)$$

Here, $K(\cdot)$ is the kernel function, where in our case, we use the Box kernel[2] $K(z)$:

$$K(z) = \frac{1}{2}, |z| \leq 1, \quad (4)$$

$$= 0, |z| > 1. \quad (5)$$

Comparing to the Gaussian kernel and other kernels, the Box kernel is simple, and computationally inexpensive. The parameter h is the bandwidth for smoothing[3].

[2] It is also known as the original Parzen windows (Parzen, 1962).

[3] In our implementation, we use the default h of the Box kernel in the *ksdensity* function in Matlab.

Now, we can derive the empirical cumulative distribution functions $\hat{F}_{X_1}(\hat{f}_1(X_1)), \hat{F}_{X_2}(\hat{f}_2(X_2)), ..., \hat{F}_{X_n}(\hat{f}_n(X_n))$ of the smoothed covariates, as well as the dependent variable y and its CDF $\hat{F}_y(\hat{f}(y))$. The empirical cumulative distribution functions are defined as:

$$\hat{F}(\nu) = \frac{1}{m}\sum_{i=1}^{m}\mathbf{I}\{x_i \leq \nu\} \tag{6}$$

where $\mathbf{I}\{\cdot\}$ is the indicator function, and ν indicates the current value that we are evaluating. Note that the above step is also known as *probability integral transform* (Diebold et al., 1997), which allows us to convert any given continuous distribution to random variables having a uniform distribution. This is of crucial importance to modeling text data: instead of using the classic bag-of-words representation that uses raw counts, we are now working with uniform marginal CDFs, which helps coping with the overfitting issue due to noise and data sparsity.

The Parametric Copula Estimation

Now that we have obtained the marginals, and then the joint distribution can be constructed by applying the copula function that models the stochastic dependencies among marginal CDFs:

$$\hat{F}(\hat{f}_1(X_1), ..., \hat{f}_1(X_n), \hat{f}(y)) \tag{7}$$
$$= C[\hat{F}_{X_1}(\hat{f}_1(X_1)), ..., \hat{F}_{X_n}(\hat{f}_n(X_n)), \hat{F}_y(\hat{f}_y(y))] \tag{8}$$

In this work, we apply the parametric Gaussian copula to model the correlations among the text features and the label. Assume x_i is the smoothed version of random variable X_i, and y is the smoothed label, we have:

$$F(x_1, ..., x_n, y) \tag{9}$$
$$= \Phi_\Sigma\Big(\Phi^{-1}[F_{x_1}(x_1)], ..., , \Phi^{-1}[F_{x_n}(x_n)], \Phi^{-1}[F_y(y)]\Big) \tag{10}$$

where Φ_Σ is the joint cumulative distribution function of a multivariate Gaussian with zero mean and Σ variance. Φ^{-1} is the inverse CDF of a standard Gaussian. In this parametric part of the model, the parameter estimation boils down to the problem of learning the covariance matrix Σ of this Gaussian copula. In this work, we perform standard maximum likelihood estimation for the Σ matrix.

To calibrate the Σ matrix, we make use of the power of randomness: using the initial Σ from MLE, we generate random samples from the Gaussian copula, and then concatenate previously generated joint of Gaussian inverse marginal CDFs with the newly generated random copula numbers, and re-estimate using MLE to derive the final adjusted Σ. Note that the final Σ matrix has to be symmetric and positive definite.

Computational Complexity

One important question regarding the proposed semiparametric Gaussian copula model is the corresponding computational complexity. This boils down to the estimation of the $\hat{\Sigma}$ matrix (Liu et al., 2012): one only needs to calculate the correlation coefficients of $n(n-1)/2$ pairs of random variables. Christensen (2005) shows that sorting and balanced binary trees can be used to calculate the correlation coefficients with complexity of $O(n \log n)$. Therefore, the computational complexity of MLE for the proposed model is $O(n \log n)$.

Efficient Approximate Inference

In this regression task, in order to perform exact inference of the conditional probability distribution $p(F_y(y)|F_{x_1}(x_1), ..., F_{x_n}(x_n))$, one needs to solve the mean response $\hat{\mathbf{E}}(F_y(y)|F_{x_1}(x_1), ..., F_{x_1}(x_1))$ from a joint distribution of high-dimensional Gaussian copula.

Assume in the simple bivariate case of Gaussian copula regression, the covariance matrix Σ is:

$$\Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ & \Sigma_{22} \end{bmatrix}$$

We can easily derive the conditional density that can be used to calculate the expected value of the CDF of the label:

$$C(F_y(y)|F_{x_1}(x_1); \Sigma) = \frac{1}{|\Sigma_{22} - \Sigma_{12}^T\Sigma_{11}^{-1}\Sigma_{12}|^{\frac{1}{2}}}$$
$$\exp\left(-\frac{1}{2}\delta^T\Big([\Sigma_{22} - \Sigma_{12}^T\Sigma_{11}^{-1}\Sigma_{12}]^{-1} - \mathbf{I}\Big)\delta\right) \tag{11}$$

where $\delta = \Phi^{-1}[F_y(y)] - \Sigma_{12}^T\Sigma_{11}^{-1}\Phi^{-1}[F_{x_1}(x_1)]$.

Unfortunately, the exact inference can be intractable in the multivariate case, and approximate inference, such as Markov Chain Monte Carlo sampling (Gelfand and Smith, 1990; Pitt et al., 2006) is often used for posterior inference. In this work, we propose an efficient sampling method to derive y given the text features — we sample $F_y(y)$ s.t. it maximizes the joint high-dimensional Gaussian copula density:

$$\hat{F_y(y)} \approx \underset{F_y(y)\in(0,1)}{\arg\max} \frac{1}{\sqrt{\det\Sigma}} \exp\left(-\frac{1}{2}\Delta^T \cdot (\Sigma^{-1} - \mathbf{I}) \cdot \Delta\right) \tag{12}$$

where

$$\Delta = \begin{pmatrix} \Phi^{-1}(F_{x_1}(x_1)) \\ \vdots \\ \Phi^{-1}(F_{x_n}(x_n)) \\ \Phi^{-1}(F_y(y)) \end{pmatrix}$$

Again, the reason why we perform approximated inference is that: exact inference in the high-dimensional Gaussian copula density is nontrivial, and might not have analytical solutions, but approximate inference using maximum density sampling from the Gaussian copula significantly relaxes the complexity of inference. Finally, to derive $\hat{y}$, the last step is to compute the inverse CDF of $\hat{F_y}(y)$.

3.3 Algorithmic Implementation

The algorithmic implementation of our semiparametric Gaussian copula text regression model is shown in Algorithm 1. Basically, the algorithm can be decomposed into four parts:

- Perform nonparametric Box kernel density estimates of the covariates and the dependent variable for smoothing.
- Calculate the empirical cumulative distribution functions of the smoothed random variables.
- Estimate the parameters (covariance Σ) of the Gaussian copula.
- Infer the predicted value of the dependent variable by sampling the Gaussian copula probability density function.

4 Datasets

We use three datasets[4] of transcribed quarterly earnings calls from the U.S. stock market, focusing on the period of the Great Recession.

The *pre-2009* dataset consists of earnings calls from the period of 2006-2008, which includes calls from the beginning of economic downturn, the outbreak of the subprime mortgage crisis, and the epidemic of collapses of large financial institutions. The *2009* dataset contains earnings calls from the year of 2009, which is a period where the credit crisis spreads globally, and the Dow Jones Industrial Average hit the lowest since the beginning of the millennium. The *post-2009* dataset includes earnings calls from the period of 2010 to 2013, which concerns the recovery of global economy. The detailed statistics is shown in Table 1.

[4]http://www.cs.cmu.edu/~yww/data/earningscalls.zip

Algorithm 1 A Semi-parametric Gaussian Copula Model Based Text Regression Algorithm

Given:
(1) training data $(X^{(tr)}, \vec{y}^{(tr)})$;
(2) testing data $(X^{(te)}, \vec{y}^{(te)})$;

Learning:
for $i = 1 \rightarrow n$ dimensions **do**
  $X_i^{(tr)'} \leftarrow BoxKDE(X_i^{(tr)}, X_i^{(tr)})$;
  $U_i^{(tr)} \leftarrow EmpiricalCDF(X_i^{(tr)'})$;
  $X_i^{(te)'} \leftarrow BoxKDE(X_i^{(tr)}, X_i^{(te)})$;
  $U_i^{(te)} \leftarrow EmpiricalCDF(X_i^{(te)'})$;
end for
$y^{(tr)'} \leftarrow BoxKDE(y^{(tr)}, y^{(tr)})$;
$v^{(tr)} \leftarrow EmpiricalCDF(y^{(tr)'})$;
$Z^{(tr)} \leftarrow GaussianInverseCDF([U^{(tr)}\ v^{(tr)}])$;
$\hat{\Sigma} \leftarrow CorrelationCoefficients(Z^{(tr)})$;
$r \leftarrow MultiVariateGaussianRandNum(0, \hat{\Sigma}, n)$;
$Z^{(tr)'} = GaussianCDF(r)$;
$\hat{\Sigma} \leftarrow CorrelationCoefficients([Z^{(tr)}\ Z^{(tr)'}])$;

Inference:
for $j = 1 \rightarrow m$ instances **do**
  $\max_j \leftarrow 0$;
  $\hat{Y}' = 0$;
  for $k = 0.01 \rightarrow 1$ **do**
    $Z^{(te)} \leftarrow GaussianInverseCDF([U^{(te)}\ k])$;
    $p_j = \frac{MultiVariateGaussianPDF(Z^{(te)}, \hat{\Sigma})}{\prod_n GaussianPDF(Z^{(te)})}$;
    if $p_j \geq \max_j$ **then**
      $\max_j = p_j$;
      $\hat{Y}' = k$;
    end if
  end for
end for
$\hat{y} \leftarrow InverseCDF(\vec{y}^{(tr)}, \hat{Y}')$;

Dataset	#Calls	#Companies	#Types	#Tokens
Pre-2009	3694	2746	371.5K	28.7M
2009	3474	2178	346.2K	26.4M
Post-2009	3726	2107	377.4K	28.6M

Table 1: Statistics of three datasets. *Types: unique words. Tokens: word tokens.*

Note that unlike the standard news corpora in NLP or the SEC-mandated financial report, Transcripts of earnings call is a very special genre of text. For example, the length of WSJ documents is typically one to three hundreds (Harman, 1995), but the averaged document length of our three earnings calls datasets is 7677. Depending on the amount of interactions in the question answering session, the complexities of the calls vary. This mixed form of formal statement and informal speech brought difficulties to machine learning algorithms.

5 Measuring Financial Risks

Volatility is an important measure of the financial risk, and in this work, we focus on predicting the future volatility following the earnings teleconfer-

ence call. For each earning call, we have a week of stock prices of the company after the day on which the earnings call is made. The *Return* of Day t is:

$$r_t = \frac{x_t}{x_{t-1}} - 1 \tag{13}$$

where x_t represents the share price of Day t, and the *Measured Stock Volatility* from Day t to $t + \tau$:

$$y_{(t,t+\tau)} = \sqrt{\frac{\sum_{i=0}^{\tau}(r_{t+i} - \bar{r})^2}{\tau}} \tag{14}$$

Using the stock prices, we can use the equations above to calculate the measured stock volatility after the earnings call, which is the standard measure of risks in finance, and the dependent variable y of our predictive task.

6 Experiments

6.1 Experimental Setup

In all experiments throughout this section, we use 80-20 train/test splits on all three datasets.

Feature sets:

We have extracted lexical, named entity, syntactic, and frame-semantics features, most of which have been shown to perform well in previous work (Xie et al., 2013). We use the unigrams and bigrams to represent lexical features, and the Stanford part-of-speech tagger (Toutanova et al., 2003) to extract the lexicalized named entity and part-of-speech features. A probabilistic frame-semantics parser, SEMAFOR (Das et al., 2010), is used to provide the FrameNet-style frame-level semantic annotations. For each of the five sets, we collect the top-100 most frequent features, and end up with a total of 500 features.

Baselines:

The baselines are standard squared-loss linear regression, linear kernel SVM, and non-linear (Gaussian) kernel SVM. They are all standard algorithms in regression problems, and have been shown to have outstanding performances in many recent text regression (Kogan et al., 2009; Chahuneau et al., 2012; Xie et al., 2013; Wang et al., 2013; Tsai and Wang, 2013). We use the Statistical Toolbox's linear regression implementation in Matlab, and LibSVM (Chang and Lin, 2011) for training and testing the SVM models. The hyperparameter C in linear SVM, and the γ and C hyperparameters in Gaussian SVM are tuned on the training set using 10-fold cross-validation. Note that since the kernel density estimation in the proposed copula model is nonparametric, and we only need to learn the Σ in the Gaussian copula, there is no hyperparameters that need to be tuned.

Evaluation Metrics:

Spearman's correlation (Hogg and Craig, 1994) and Kendall's tau (Kendall, 1938) have been widely used in many regression problems in NLP (Albrecht and Hwa, 2007; Yogatama et al., 2011; Wang et al., 2013; Tsai and Wang, 2013), and here we use them to measure the quality of predicted values $\hat{\mathbf{y}}$ by comparing to the vector of ground truth $\mathbf{y}$. In contrast to Pearson's correlation, Spearman's correlation has no assumptions on the relationship of the two measured variables. Kendall's tau is a nonparametric statistical metric that have shown to be inexpensive, robust, and representation independent (Lapata, 2006). We also use paired two-tailed t-test to measure the statistical significance between the best and the second best approaches.

6.2 Comparing to Various Baselines

In the first experiment, we compare the proposed semiparametric Gaussian copula regression model to three baselines on three datasets with all features. The detailed results are shown in Table 2. On the *pre-2009* dataset, we see that the linear regression and linear SVM perform reasonably well, but the Gaussian kernel SVM performs less well, probably due to overfitting. The copula model outperformed all three baselines by a wide margin on this dataset with both metrics. Similar performances are also obtained in the *2009* dataset, where the result of linear SVM baseline falls behind. On the *post-2009* dataset, none of results from the linear and non-linear SVM models can match up with the linear regression model, but our proposed copula model still improves over all baselines by a large margin. Comparing to second-best approaches, all improvements obtained by the copula model are statistically significant.

6.3 Varying the Amount of Training Data

To understand the learning curve of our proposed copula regression model, we use the 25%, 50%, 75% subsets from the training data, and evaluate all four models. Figure 1 shows the evaluation results. From the experiments on the *pre-2009* dataset, we see that when the amount of training data is small (25%), both SVM models have obtained very impressive results. This is not surprising at all, because as max-margin models, soft-margin SVM only needs a handful of examples that come with nonvanishing coefficients (support vectors) to find a reasonable margin. When in-

Method	Pre-2009		2009		Post-2009	
	Spearman	Kendall	Spearman	Kendall	Spearman	Kendall
linear regression:	0.377	0.259	0.367	0.252	0.314	0.216
linear SVM:	0.364	0.249	0.242	0.167	0.132	0.091
Gaussian SVM:	0.305	0.207	0.280	0.192	0.152	0.104
Gaussian copula:	**0.425***	**0.315***	**0.422***	**0.310***	**0.375***	**0.282***

Table 2: Comparing the learning algorithms on three datasets with all features. The best result is highlighted in **bold**. * indicates $p < .001$ comparing to the second best result.

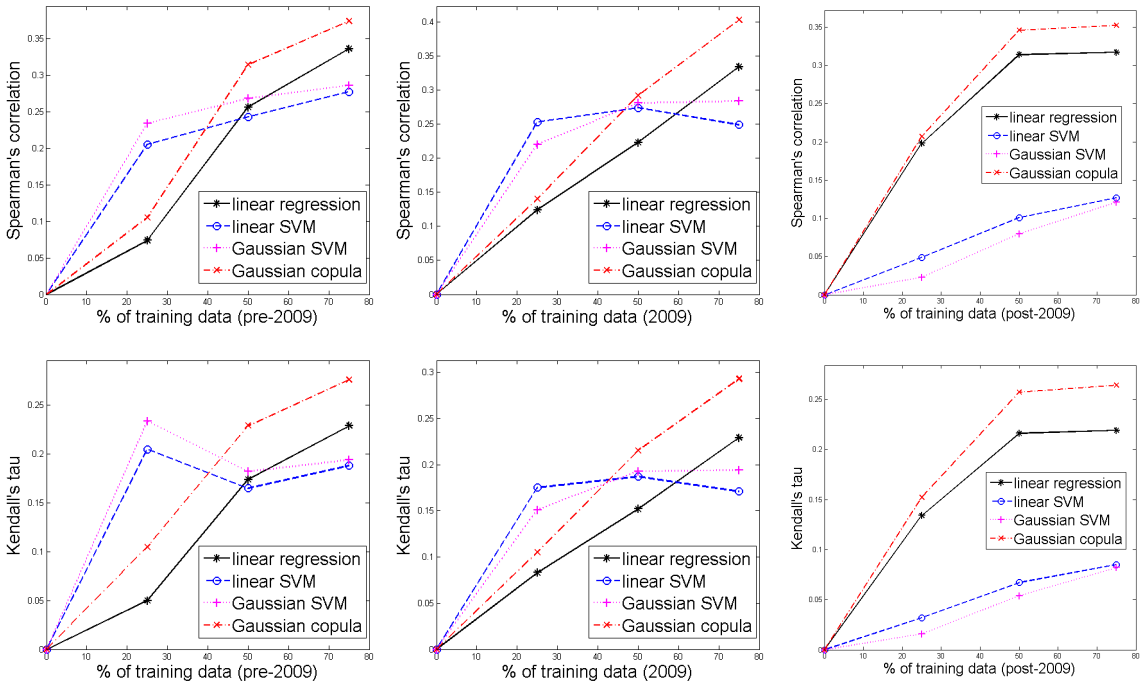


Figure 1: Varying the amount of training data. Left column: pre-2009 dataset. Middle column: 2009 dataset. Right column: post-2009 dataset. Top row: Spearman's correlation. Bottom row: Kendall's tau.

creasing the amount of training data to 50%, we do see the proposed copula model catches up quickly, and lead all baseline methods undoubtably at 75% training data. On the *2009* dataset, we observe very similar patterns. Interestingly, the proposed copula regression model has dominated all methods for both metrics throughout all proportions of the "post-2009" earnings calls dataset, where instead of financial crisis, the economic recovery is the main theme. In contrast to the previous two datasets, both linear and non-linear SVMs fail to reach reasonable performances on this dataset.

6.4 Varying the Amount of Features

Finally, we investigate the robustness of the proposed semiparametric Gaussian copula regression model by varying the amount of features in the covariate space. To do this, we sample equal amount of features from each feature set, and concatenate them into a feature vector. When increasing the amount of total features from 100 to 400, the results are shown in Figure 2. On the *pre-2009* dataset, we see that the gaps between the best-perform copula model and the second-best linear regression model are consistent throughout all feature sizes. On the *2009* dataset, we see that the performance of Gaussian copula is aligned with the linear regression model in terms of Spearman's correlation, where the former seems to perform better in terms of Kendall's tau. Both linear and non-linear SVM models do not have any advantages over the proposed approach. On the *post-2009* dataset that concerns economic growth and recovery, the boundaries among all methods are very clear. The Spearman's correlation for both SVM baselines is less than 0.15 throughout all settings, but copula model is able to achieve 0.4 when using 400 features. The improvements of copula

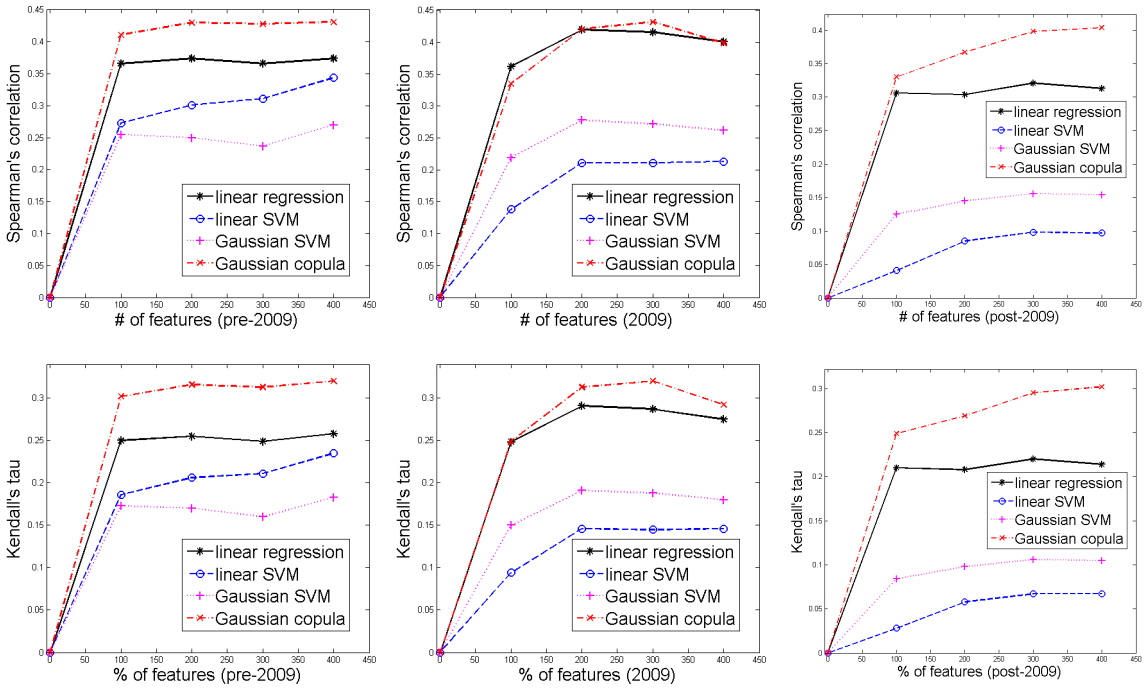


Figure 2: Varying the amount of features. Left column: pre-2009 dataset. Middle column: 2009 dataset. Right column: post-2009 dataset. Top row: Spearman's correlation. Bottom row: Kendall's tau.

Pre-2009	2009	Post-2009
2008/CD	2008	first_quarter
2008	million/CD	revenue/NN
third_quarter	2008/CD	revenue
third	million	quarter_of
third/JJ	million_in	compared_to
the_third	the_fourth	million_in
million/CD	fourth_quarter	Peter/PERSON
capital	fourth	call
million	fourth/JJ	first/JJ
FE_Trajector_entity	$/$	million/CD

Table 3: Top-10 features that have positive correlations with stock volatility in three datasets.

model over squared loss linear regression model are increasing, when working with larger feature spaces.

6.5 Qualitative Analysis

Like linear classifiers, by "opening the hood" to the Gaussian copula regression model, one can examine features that exhibit high correlations with the dependent variable. Table 3 shows the top features that are positively correlated with the future stock volatility in the three datasets. On the top features from the "pre-2009" dataset, which primarily (82%) includes calls from 2008, we can clearly observe that the word "2008" has strong correlation with the financial risks. Interestingly, the phrase "third quarter" and its variations, not only play an important role in the model, but also highly correlated to the timeline of the financial crisis: the Q3 of 2008 is a critical period in the recession, where Lehman Brothers falls on the Sept. 15 of 2008, filing $613 billion of debt — the biggest bankruptcy in U.S. history (Mamudi, 2008). This huge panic soon broke out in various financial institutions in the Wall Street. On the top features from "2009" dataset, again, we see the word "2008" is still prominent in predicting financial risks, indicating the hardship and extended impacts from the center of the economic crisis. After examining the transcripts, we found sentences like: "*...our specialty lighting business that we discontinued in the fourth quarter of 2008...*", "*...the exception of fourth quarter revenue which was $100,000 below our guidance target...*", and "*...to address changing economic conditions and their impact on our operations, in the fourth quarter we took the painful but prudent step of decreasing our headcount by about 5%...*", showing the crucial role that Q4 of 2008 plays in 2009 earnings calls. Interestingly, after the 2008-2009 crisis, in the recovery period, we have observed new words like "revenue", indicating the "back-to-normal" trend of financial environment, and new features that predict financial volatility.

7 Discussions

In the experimental section, we notice that the proposed semiparametric Gaussian copula model has obtained promising results in various setups on three datasets in this text regression task. The

main questions we ask are: how is the proposed model different from standard text regression/classification models? What are the advantages of copula-based models, and what makes it perform so well?

One advantage we see from the copula model is that it does not require any assumptions on the marginal distributions. For example, in latent Dirichlet allocation (Blei et al., 2003), the topic proportion of a document is always drawn from a $Dirichlet(\alpha)$ distribution. This is rather restricted, because the possible shapes from a $K-1$ simplex of Dirichlet is always limited in some sense. In our copula model, instead of using some priors, we just calculate the empirical cumulative distribution function of the random variables, and model the correlation among them. This is extremely practical, because in many natural language processing tasks, we often have to deal with features that are extracted from many different domains and signals. By applying the *Probability Integral Transform* to raw features in the copula model, we essentially avoid comparing apples and oranges in the feature space, which is a common problem in bag-of-features models in NLP.

The second hypothesis is about the semiparametirc parameterization, which contains the nonparametric kernel density estimation and the parametric Gaussian copula regression components. The benefit of a semiparametric model is that here we are not interested in performing completely nonparametric estimations, where the infinite dimensional parameters might bring intractability. In contrast, by considering the semiparametric case, we not only obtain some expressiveness from the nonparametric models, but also reduce the complexity of the task: we are only interested in the finite-dimensional components Σ in the Gaussian copula with $O(n \log n)$ complexity, which is not as computationally difficult as the completely nonparametric cases. Also, by modeling the marginals and their correlations seperately, our approach is cleaner, easy-to-understand, and allows us to have more flexibility to model the uncertainty of data. Our pilot experiment also aligns with our hypothesis: when not performing the kernel density estimation part for smoothing out the marginal distributions, the performances dropped significantly when sparser features are included.

The third advantage we observe is the power of modeling the covariance of the random variables. Traditionally, in statistics, independent and identically distributed (i.i.d) assumptions among the instances and the random variables are often used in various models, such that the correlations among the instances or the variables are often ignored. However, this might not be practical at all: in image processing, the "cloud" pixel of a pixel showing the blue sky of a picture are more likelihood to co-occur in the same picture; in natural language processing, the word "mythical" is more likely to co-occur with the word "unicorn", rather than the word "popcorn". Therefore, by modeling the correlations among marginal CDFs, the copula model has gained the insights on the dependency structures of the random variables, and thus, the performance of the regression task is boosted.

In the future, we plan to apply the proposed approach to large datasets where millions of features and millions of instances are involved. Currently we have not experienced the difficulty when estimating the Gaussian copula model, but parallel methods might be needed to speedup learning when significantly more marginal CDFs are involved. The second issue is about overfitting. We see that when features are rather noisy, we might need to investigate regularized copula models to avoid this. Finally, we plan to extend the proposed approach to text classification and structured prediction problems in NLP.

8 Conclusion

In this work, we have demonstrated that the more complex quarterly earnings calls can also be used to predict the measured volatility of the stocks in the limited future. We propose a novel semiparametric Gausian copula regression approach that models the dependency structure of the language in the earnings calls. Unlike traditional bag-of-features models that work discrete features from various signals, we perform kernel density estimation to smooth out the distribution, and use probability integral transform to work with CDFs that are uniform. The copula model deals with marginal CDFs and the correlation among them separately, in a cleaner manner that is also flexible to parameterize. Focusing on the three financial crisis related datasets, the proposed model significantly outperform the standard linear regression method in statistics and strong discriminative support vector regression baselines. By varying the size of the training data and the dimensionality of the covariates, we have demonstrated that our proposed model is relatively robust across different parameter settings.

Acknowledgement

We thank Alex Smola, Barnabás Póczos, Sam Thomson, Shoou-I Yu, Zi Yang, and anonymous reviewers for their useful comments.

References

Joshua Albrecht and Rebecca Hwa. 2007. Regression for sentence-level mt evaluation with pseudo references. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*.

David Blei, Andrew Ng, and Michael Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*.

Johan Bollen, Huina Mao, and Xiaojun Zeng. 2011. Twitter mood predicts the stock market. *Journal of Computational Science*.

Belinda Camiciottoli. 2010. Earnings calls: Exploring an emerging financial reporting genre. *Discourse & Communication*.

Victor Chahuneau, Kevin Gimpel, Bryan R Routledge, Lily Scherlis, and Noah A Smith. 2012. Word salad: Relating food prices and descriptions. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*.

Xiaohong Chen and Yanqin Fan. 2006. Estimation of copula-based semiparametric time series models. *Journal of Econometrics*.

David Christensen. 2005. Fast algorithms for the calculation of kendalls τ. *Computational Statistics*.

Kenneth Church and William Gale. 1995. Poisson mixtures. *Natural Language Engineering*.

Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A Smith. 2010. Probabilistic frame-semantic parsing. In *Human language technologies: The 2010 annual conference of the North American chapter of the association for computational linguistics*.

Francis X Diebold, Todd A Gunther, and Anthony S Tay. 1997. Evaluating density forecasts.

Carsten Eickhoff, Arjen P. de Vries, and Kevyn Collins-Thompson. 2013. Copulas for information retrieval. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Jacob Eisenstein, Brendan O'Connor, Noah A Smith, and Eric P Xing. 2010. A latent variable model for geographic lexical variation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*.

Pui Cheong Fung, Xu Yu, and Wai Lam. 2003. Stock prediction: Integrating text mining approach using real-time news. In *Proceedings of IEEE International Conference on Computational Intelligence for Financial Engineering*.

Alan Gelfand and Adrian Smith. 1990. Sampling-based approaches to calculating marginal densities. *Journal of the American statistical association*.

Christian Genest and Anne-Catherine Favre. 2007. Everything you always wanted to know about copula modeling but were afraid to ask. *Journal of Hydrologic Engineering*.

Zoubin Ghahramani, Barnabás Póczos, and Jeff Schneider. 2012. Copula-based kernel dependency measures. In *Proceedings of the 29th International Conference on Machine Learning*.

Fang Han, Tuo Zhao, and Han Liu. 2012. Coda: High dimensional copula discriminant analysis. *Journal of Machine Learning Research*.

Donna Harman. 1995. Overview of the second text retrieval conference (trec-2). *Information Processing & Management*.

Robert V Hogg and Allen Craig. 1994. Introduction to mathematical statistics.

Harry Joe. 1997. *Multivariate models and dependence concepts*.

Mahesh Joshi, Dipanjan Das, Kevin Gimpel, and Noah A Smith. 2010. Movie reviews and revenues: An experiment in text regression. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Maurice Kendall. 1938. A new measure of rank correlation. *Biometrika*.

Shimon Kogan, Dimitry Levin, Bryan Routledge, Jacob Sagi, and Noah Smith. 2009. Predicting risk from financial reports with regression. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

John Lafferty and David Blei. 2005. Correlated topic models. In *Advances in neural information processing systems*.

Mirella Lapata. 2006. Automatic evaluation of information ordering: Kendall's tau. *Computational Linguistics*.

Han Liu, Fang Han, Ming Yuan, John Lafferty, and Larry Wasserman. 2012. High-dimensional semiparametric gaussian copula graphical models. *The Annals of Statistics*.

David Lopez-paz, Jose M Hernández-lobato, and Ghahramani Zoubin. 2013. Gaussian process vine copulas for multivariate dependence. In *Proceedings of the 30th International Conference on Machine Learning*.

Sam Mamudi. 2008. *Lehman folds with record $613 billion debt*. MarketWatch.com.

Guido Masarotto and Cristiano Varin. 2012. Gaussian copula marginal regression. *Electronic Journal of Statistics*.

Roger B Nelsen. 1999. *An introduction to copulas*. Springer Verlag.

Rahul A Parsa and Stuart A Klugman. 2011. Copula regression. *Variance Advancing and Science of Risk*.

Emanuel Parzen. 1962. On estimation of a probability density function and mode. *The annals of mathematical statistics*.

Michael Pitt, David Chan, and Robert Kohn. 2006. Efficient bayesian inference for gaussian copula regression models. *Biometrika*.

McKay Price, James Doran, David Peterson, and Barbara Bliss. 2012. Earnings conference calls and stock returns: The incremental informativeness of textual tone. *Journal of Banking & Finance*.

Joseph Reisinger, Austin Waters, Bryan Silverthorn, and Raymond J Mooney. 2010. Spherical topic models. In *Proceedings of the 27th International Conference on Machine Learning*.

Berthold Schweizer and Abe Sklar. 1983. *Probabilistic metric spaces*.

Claude Shannon. 1948. A mathematical theory of communication. In *The Bell System Technical Journal*.

Abe Sklar. 1959. *Fonctions de répartition à n dimensions et leurs marges*. Université Paris 8.

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

Ming-Feng Tsai and Chuan-Ju Wang. 2013. Risk ranking from financial reports. In *Advances in Information Retrieval*.

William Yang Wang, Elijah Mayfield, Suresh Naidu, and Jeremiah Dittmar. 2012. Historical analysis of legal opinions with a sparse mixed-effects latent variable model. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*.

Chuan-Ju Wang, Ming-Feng Tsai, Tse Liu, and Chin-Ting Chang. 2013. Financial sentiment analysis for risk prediction. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*.

Rongjing Xiang and Jennifer Neville. 2013. Collective inference for network data with copula latent markov networks. In *Proceedings of the sixth ACM international conference on Web search and data mining*.

Boyi Xie, Rebecca J. Passonneau, Leon Wu, and Germán G. Creamer. 2013. Semantic frames to predict stock price movement. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

Dani Yogatama, Michael Heilman, Brendan O'Connor, Chris Dyer, Bryan R Routledge, and Noah A Smith. 2011. Predicting a scientific community's response to an article. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Xue Zhang, Hauke Fuehres, and Peter A Gloor. 2011. Predicting stock market indicators through twitter "i hope it is not as bad as i fear". *Procedia-Social and Behavioral Sciences*.

Polylingual Tree-Based Topic Models for Translation Domain Adaptation

Yuening Hu[†]
Computer Science
University of Maryland
ynhu@cs.umd.edu

Ke Zhai[†]
Computer Science
University of Maryland
zhaike@cs.umd.edu

Vladimir Eidelman
FiscalNote Inc.
Washington DC
vlad@fiscalnote.com

Jordan Boyd-Graber
iSchool and UMIACS
University of Maryland
jbg@umiacs.umd.edu

Abstract

Topic models, an unsupervised technique for inferring translation domains improve machine translation quality. However, previous work uses only the source language and completely ignores the target language, which can disambiguate domains. We propose new polylingual tree-based topic models to extract domain knowledge that considers both source and target languages and derive three different inference schemes. We evaluate our model on a Chinese to English translation task and obtain up to 1.2 BLEU improvement over strong baselines.

1 Introduction

Probabilistic topic models (Blei and Lafferty, 2009), exemplified by *latent Dirichlet allocation* (Blei et al., 2003, LDA), are one of the most popular statistical frameworks for navigating large unannotated document collections. Topic models discover—without any supervision—the primary themes presented in a dataset: the namesake topics.

Topic models have two primary applications: to aid human exploration of corpora (Chang et al., 2009) or serve as a low-dimensional representation for downstream applications. We focus on the second application, which has been fruitful for computer vision (Li Fei-Fei and Perona, 2005), computational biology (Perina et al., 2010), and information retrieval (Kataria et al., 2011).

In particular, we use topic models to aid *statistical machine translation* (Koehn, 2009, SMT). Modern machine translation systems use millions of examples of translations to learn translation rules. These systems work best when the training corpus has consistent genre, register, and topic. Systems that are robust to systematic variation in the training set are said to exhibit *domain adaptation*.

† indicates equal contributions.

As we review in Section 2, topic models are a promising solution for automatically discovering domains in machine translation corpora. However, past work either relies solely on monolingual source-side models (Eidelman et al., 2012; Hasler et al., 2012; Su et al., 2012), or limited modeling of the target side (Xiao et al., 2012). In contrast, machine translation uses inherently multilingual data: an SMT system must translate a phrase or sentence from a *source* language to a different *target* language, so existing applications of topic models (Eidelman et al., 2012) are wilfully ignoring available information on the target side that could aid domain discovery.

This is not for a lack of multilingual topic models. Topic models bridge the chasm between languages using document connections (Mimno et al., 2009), dictionaries (Boyd-Graber and Resnik, 2010), and word alignments (Zhao and Xing, 2006). In Section 2, we review these models for discovering topics in multilingual datasets and discuss how they can improve SMT.

However, no models combine multiple bridges between languages. In Section 3, we create a model—the polylingual tree-based topic models (ptLDA)—that uses information from both external dictionaries and document alignments simultaneously. In Section 4, we derive both MCMC and variational inference for this new topic model.

In Section 5, we evaluate our model on the task of SMT using aligned datasets. We show that ptLDA offers better domain adaptation than other topic models for machine translation. Finally, in Section 6, we show how these topic models improve SMT with detailed examples.

2 Topic Models for Machine Translation

Before considering past approaches using topic models to improve SMT, we briefly review lexical weighting and domain adaptation for SMT.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1166–1176,
Baltimore, Maryland, USA, June 23-25 2014.

2.1 Statistical Machine Translation

Statistical machine translation casts machine translation as a probabilistic process (Koehn, 2009). For a parallel corpus of aligned source and target sentences $(\mathcal{F}, \mathcal{E})$, a phrase $\bar{f} \in \mathcal{F}$ is translated to a phrase $\bar{e} \in \mathcal{E}$ according to a distribution $p_w(\bar{e}|\bar{f})$. One popular method to estimate the probability $p_w(\bar{e}|\bar{f})$ is via lexical weighting features.

Lexical Weighting In phrase-based SMT, lexical weighting features estimate the phrase pair quality by combining lexical translation probabilities of words in a phrase (Koehn et al., 2003). Lexical conditional probabilities $p(e|f)$ are maximum likelihood estimates from relative lexical frequencies $c(f,e)/\sum_e c(f,e)$, where $c(f,e)$ is the count of observing lexical pair (f,e) in the training dataset. The phrase pair probabilities $p_w(\bar{e}|\bar{f})$ are the normalized product of lexical probabilities of the aligned word pairs within that phrase pair (Koehn et al., 2003). In Section 2.2, we create topic-specific lexical weighting features.

Cross-Domain SMT A SMT system is usually trained on documents with the same genre (e.g., sports, business) from a similar style (e.g., newswire, blog-posts). These are called *domains*. Translations within one domain are better than translations across domains since they vary dramatically in their word choices and style. A correct translation in one domain may be inappropriate in another domain. For example, "潜水" in a newspaper usually means "underwater diving". On social media, it means a non-contributing "lurker".

Domain Adaptation for SMT Training a SMT system using diverse data requires *domain adaptation*. Early efforts focus on building separate models (Foster and Kuhn, 2007) and adding features (Matsoukas et al., 2009) to model domain information. Chiang et al. (2011) combine these approaches by directly optimizing genre and collection features by computing separate translation tables for each domain.

However, these approaches treat domains as hand-labeled, constant, and known *a priori*. This setup is at best expensive and at worst infeasible for large data. Topic models provide a solution where domains can be automatically induced from raw data: treat each topic as a domain.[1]

[1]Henceforth we will use the term "topic" and "domain" interchangeably: "topic" to refer to the concept in topic models and "domain" to refer to SMT corpora.

2.2 Inducing Domains with Topic Models

Topic models take the number of topics K and a collection of documents as input, where each document is a bag of words. They output two distributions: a distribution over topics for each document d; and a distribution over words for each topic. If each topic defines a SMT domain, the document's topic distribution is a soft domain assignment for that document.

Given the soft domain assignments, Eidelman et al. (2012) extract lexical weighting features conditioned on the topics, optimizing feature weights using the *Margin Infused Relaxed Algorithm* (Crammer et al., 2006, MIRA). The topics come from source documents *only* and create topic-specific lexical weights from the per-document topic distribution $p(k \mid d)$. The lexical probability conditioned on the topic is expected count $e_k(e,f)$ of a word translation pair under topic k,

$$\hat{c}_k(e,f) = \textstyle\sum_d p(k|d) c_d(e,f), \qquad (1)$$

where $c_d(\bullet)$ is the number of occurrences of the word pair in document d. The lexical probability conditioned on topic k is the unsmoothed probability estimate of those expected counts

$$p_w(e|f;k) = \tfrac{\hat{c}_k(e,f)}{\sum_e \hat{c}_k(e,f)}, \qquad (2)$$

from which we can compute the phrase pair probabilities $p_w(\bar{e}|\bar{f};k)$ by multiplying the lexical probabilities and normalizing as in Koehn et al. (2003).

For a test document d, the document topic distribution $p(k \mid d)$ is inferred based on the topics learned from training data. The feature value of a phrase pair $(\bar{e}, \bar{f})$ is

$$f_k(\bar{e}|\bar{f}) = -\log\left\{p_w(\bar{e}|\bar{f};k) \cdot p(k|d)\right\}, \qquad (3)$$

a combination of the topic dependent lexical weight and the topic distribution of the document, from which we extract the phrase. Eidelman et al. (2012) compute the resulting model score by combining these features in a linear model with other standard SMT features and optimizing the weights.

Conceptually, this approach is just reweighting examples. The probability of a topic given a document is never zero. Every translation observed in the training set will contribute to $p_k(e|f)$; many of the expected counts, however, will be less than one. This obviates the explicit smoothing used in other domain adaptation systems (Chiang et al., 2011).

We adopt this framework in its entirety. Our contribution are topics that capture *multilingual* information and thus better capture the domains in the parallel corpus.

2.3 Beyond Vanilla Topic Models

Eidelman et al. (2012) ignore a wealth of information that could improve topic models and help machine translation. Namely, they only use monolingual data from the source language, ignoring all target-language data and available lexical semantic resources between source and target languages.

Different complement each other to reduce ambiguity. For example, "木马" in a Chinese document can be either "hobbyhorse" in a children's topic, or "Trojan virus" in a technology topic. A short Chinese context obscures the true topic. However, these terms are unambiguous in English, revealing the true topic.

While vanilla topic models (LDA) can only be applied to monolingual data, there are a number of topic models for parallel corpora: Zhao and Xing (2006) assume aligned word pairs share same topics; Mimno et al. (2009) connect different languages through comparable documents. These models take advantage of word or document *alignment information* and infer more robust topics from the aligned dataset.

On the other hand, *lexical information* can induce topics from multilingual corpora. For instance, orthographic similarity connects words with the same meaning in related languages (Boyd-Graber and Blei, 2009), and dictionaries are a more general source of information on which words share meaning (Boyd-Graber and Resnik, 2010).

These two approaches are not mutually exclusive, however; they reveal different connections across languages. In the next section, we combine these two approaches into a polylingual tree-based topic model.

3 Polylingual Tree-based Topic Models

In this section, we bring existing tree-based topic models (Boyd-Graber et al., 2007, tLDA) and polylingual topic models (Mimno et al., 2009, pLDA) together and create the polylingual tree-based topic model (ptLDA) that incorporates both word-level correlations and document-level alignment information.

Word-level Correlations Tree-based topic models incorporate the correlations between words by encouraging words that appear together in a **concept** to have similar probabilities given a topic. These concepts can come from WordNet (Boyd-Graber and Resnik, 2010), domain experts (Andrzejewski et al., 2009), or user constrains (Hu et al., 2013). When we gather concepts from bilingual resources, these concepts can connect different languages. For example, if a bilingual dictionary defines "电脑" as "computer", we combine these words in a concept.

We organize the vocabulary in a tree structure based on these concepts (Figure 1): words in the same concept share a common parent node, and then that concept becomes one of many children of the root node. Words that are not in any concept—**uncorrelated words**—are directly connected to the root node. We call this structure the **tree prior**.

When this tree serves as a prior for topic models, words in the same concept are correlated in topics. For example, if "电脑" has high probability in a topic, so will "computer", since they share the same parent node. With the tree priors, each topic is no longer a distribution over word types, instead, it is a distribution over paths, and each path is associated with a word type. The same word could appear in multiple paths, and each path represents a unique sense of this word.

Document-level Alignments Lexical resources connect languages and help guide the topics. However, these resources are sometimes brittle and may not cover the whole vocabulary. Aligned document pairs provide a more corpus-specific, flexible association across languages.

Polylingual topic models (Mimno et al., 2009) assume that the aligned documents in different languages share the same topic distribution and each language has a unique topic distribution over its word types. This level of connection between languages is flexible: instead of requiring the exact matching on words and sentences, only a coarse document alignment is necessary, as long as the documents discuss the same topics.

Combine Words and Documents We propose polylingual tree-based topic models (ptLDA), which connect information across different languages by incorporating both word correlation (as in tLDA) and document alignment information (as in pLDA). We initially assume a given tree structure, deferring the tree's provenance to the end of this section.

Generative Process As in LDA, each word token is associated with a topic. However, tree-based topic models introduce an additional step of selecting a concept in a topic responsible for generating each word token. This is represented by a path $y_{d,n}$ through the topic's tree.

The probability of a path in a topic depends on the transition probabilities in a topic. Each concept i in topic k has a distribution over its children nodes is governed by a Dirichlet prior: $\pi_{k,i} \sim \text{Dir}(\beta_i)$. Each path ends in a word (i.e., a leaf node) and the probability of a path is the product of all of the transitions between topics it traverses. Topics have correlations over words because the Dirichlet parameters can encode positive or negative correlations (Andrzejewski et al., 2009).

With these correlated in topics in hand, the generation of documents are very similar to LDA. For every document d, we first sample a distribution over topics θ_d from a Dirichlet prior $\text{Dir}(\alpha)$. For every token in the documents, we first sample a topic z_{dn} from the multinomial distribution θ_d, and then sample a path y_{dn} along the tree according to the transition distributions specified by topic z_{dn}. Because every path y_{dn} leads to a word w_{dn} in language l_{dn}, we append the sampled word w_{dn} to document $d_{l_{dn}}$. Aligned documents have words in both languages; monolingual documents only have words in a single language.

The full generative process is:

for topic $k \in 1, \cdots, K$ **do**
  for each internal node n_i **do**
    draw a distribution $\pi_{ki} \sim \text{Dir}(\beta_i)$
for document set $d \in 1, \cdots, D$ **do**
  draw a distribution $\theta_d \sim \text{Dir}(\alpha)$
  for each word in documents d **do**
    choose a topic $z_{dn} \sim \text{Mult}(\theta_d)$
    sample a path y_{dn} with probability $\prod_{(i,j) \in y_{dn}} \pi_{z_{dn},i,j}$
    y_{dn} leads to word w_{dn} in language l_{dn}
    append token w_{dn} to document $d_{l_{dn}}$

If we use a flat symmetric Dirichlet prior instead of the tree prior, we recover pLDA; and if all documents are monolingual (i.e., with distinct distributions over topics θ), we recover tLDA. ptLDA connects different languages on both the word level (using the word correlations) and the document level (using the document alignments). We compare these models' machine translation performance in Section 5.

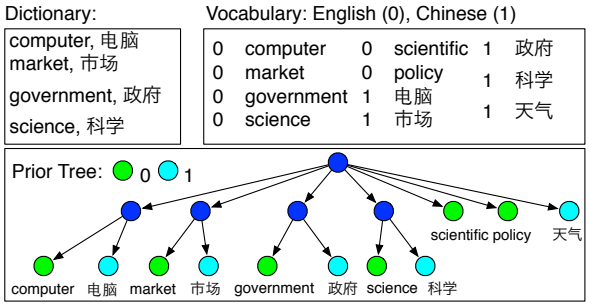


Figure 1: An example of constructing a prior tree from a bilingual dictionary: word pairs with the same meaning but in different languages are concepts; we create a common parent node to group words in a concept, and then connect to the root; uncorrelated words are connected to the root directly. Each topic uses this tree structure as a prior.

Build Prior Tree Structures One remaining question is the source of the word-level connections across languages for the tree prior. We consider two resources to build trees that correlate words across languages. The first are a multilingual dictionaries (*dict*), which match words with the same meaning in different languages together. These relations between words are used as the concepts in the prior tree (Figure 1).

In addition, we extract the word alignments from aligned sentences in a parallel corpus. The word pairs define concepts for the prior tree (*align*). We use both resources for our models (denoted as ptLDA-*dict* and ptLDA-*align*) in our experiments (Section 5) and show that they yield comparable performance in SMT.

4 Inference

Inference of probabilistic models discovers the posterior distribution over latent variables. For a collection of D documents, each of which contains N_d number of words, the latent variables of ptLDA are: transition distributions $\boldsymbol{\pi}_{ki}$ for every topic k and internal node i in the prior tree structure; multinomial distributions over topics $\boldsymbol{\theta}_d$ for every document d; topic assignments z_{dn} and path y_{dn} for the n^{th} word w_{dn} in document d. The joint distribution of polylingual tree-based topic models is

$$p(\boldsymbol{w}, \boldsymbol{z}, \boldsymbol{y}, \boldsymbol{\theta}, \boldsymbol{\pi}; \alpha, \beta) = \prod_k \prod_i p(\boldsymbol{\pi}_{ki}|\beta_i) \cdot \prod_d p(\boldsymbol{\theta}_d|\alpha) \cdot \prod_d \prod_n p(z_{dn}|\boldsymbol{\theta}_d) \cdot \prod_d \prod_n \big(p(y_{dn}|z_{dn}, \boldsymbol{\pi}) p(w_{dn}|y_{dn})\big). \quad (4)$$

Exact inference is intractable, so we turn to ap-

proximate posterior inference to discover the latent variables that best explain our data. Two widely used approximation approaches are *Markov chain Monte Carlo* (Neal, 2000, MCMC) and *variational Bayesian inference* (Blei et al., 2003, VB). Both frameworks produce good approximations of the posterior mode (Asuncion et al., 2009). In addition, Mimno et al. (2012) propose hybrid inference that takes advantage of parallelizable variational inference for global variables (Wolfe et al., 2008) while enjoying the sparse, efficient updates for local variables (Neal, 1993). In the rest of this section, we discuss all three methods in turn.

We explore multiple inference schemes because while all of these methods optimize likelihood because they might give different results on the translation task.

4.1 Markov Chain Monte Carlo Inference

We use a collapsed Gibbs sampler for tree-based topic models to sample the path y_{dn} and topic assignment z_{dn} for word w_{dn},

$$\begin{aligned} p(z_{dn} = k, y_{dn} = s | \neg \boldsymbol{z}_{dn}, \neg \boldsymbol{y}_{dn}, \boldsymbol{w}; \alpha, \boldsymbol{\beta}) \\ \propto \mathbb{I}\left[\Omega(s) = w_{dn}\right] \cdot \frac{N_{k|d} + \alpha}{\sum_{k'} (N_{k'|d} + \alpha)} \\ \cdot \prod_{i \to j \in s} \frac{N_{i \to j|k} + \beta_{i \to j}}{\sum_{j'} (N_{i \to j'|k} + \beta_{i \to j'})}, \end{aligned}$$

where $\Omega(s)$ represents the word that path s leads to, $N_{k|d}$ is the number of tokens assigned to topic k in document d and $N_{i \to j|k}$ is the number of times edge $i \to j$ in the tree assigned to topic k, excluding the topic assignment z_{dn} and its path y_{dn} of current token w_{dn}. In practice, we sample the latent variables using efficient sparse updates (Yao et al., 2009; Hu and Boyd-Graber, 2012).

4.2 Variational Bayesian Inference

Variational Bayesian inference approximates the posterior distribution with a simplified *variational distribution* q over the latent variables: document topic proportions θ, transition probabilities π, topic assignments z, and path assignments y.

Variational distributions typically assume a mean-field distribution over these latent variables, removing all dependencies between the latent variables. We follow this assumption for the transition probabilities $q(\boldsymbol{\pi} \mid \boldsymbol{\lambda})$ and the document topic proportions $q(\boldsymbol{\theta} \mid \boldsymbol{\gamma})$; both are variational Dirichlet distributions. However, due to the tight coupling between the path and topic variables, we must model this joint distribution as one multinomial, $q(\boldsymbol{z}, \boldsymbol{y} \mid \boldsymbol{\phi})$. If word token w_{dn} has K topics and S paths, it has a $K * S$ length variational multinomial ϕ_{dnks}, which represents the probability that the word takes path s in topic k. The complete variational distribution is

$$\begin{aligned} q(\boldsymbol{\theta}, \boldsymbol{\pi}, \boldsymbol{z}, \boldsymbol{y} | \boldsymbol{\gamma}, \boldsymbol{\lambda}, \boldsymbol{\phi}) = \textstyle\prod_d q(\boldsymbol{\theta}_d | \boldsymbol{\gamma}_d) \cdot \\ \textstyle\prod_k \prod_i q(\boldsymbol{\pi}_{ki} | \lambda_{ki}) \cdot \prod_d \prod_n q(z_{dn}, y_{dn} | \boldsymbol{\phi}_{dn}). \end{aligned} \tag{5}$$

Our goal is to find the variational distribution q that is closest to the true posterior, as measured by the *Kullback-Leibler* (KL) divergence between the true posterior p and variational distribution q. This induces an "evidence lower bound" (ELBO, $\mathcal{L}$) as a function of a variational distribution q: $\mathcal{L} =$

$$\begin{aligned} &\mathbb{E}_q[\log p(\boldsymbol{w}, \boldsymbol{z}, \boldsymbol{y}, \boldsymbol{\theta}, \boldsymbol{\pi})] - \mathbb{E}_q[\log q(\boldsymbol{\theta}, \boldsymbol{\pi}, \boldsymbol{z}, \boldsymbol{y})] \\ &= \textstyle\sum_k \sum_i \mathbb{E}_q[\log p(\boldsymbol{\pi}_{ki} | \beta_i)] \\ &+ \textstyle\sum_d \mathbb{E}_q[\log p(\boldsymbol{\theta}_d | \alpha)] \\ &+ \textstyle\sum_d \sum_n \mathbb{E}_q[\log p(z_{dn}, y_{dn} | \boldsymbol{\theta}_d, \boldsymbol{\pi}) p(w_{dn} | y_{dn})] \\ &+ \mathbb{H}[q(\boldsymbol{\theta})] + \mathbb{H}[q(\boldsymbol{\pi})] + \mathbb{H}[q(\boldsymbol{z}, \boldsymbol{y})], \end{aligned} \tag{6}$$

where $\mathbb{H}[\bullet]$ represents the entropy of a distribution. Optimizing $\mathcal{L}$ using coordinate descent provides the following updates:

$$\begin{aligned} \phi_{dnkt} &\propto \exp\{\Psi(\gamma_{dk}) - \Psi(\textstyle\sum_k \gamma_{dk}) \\ &\quad + \textstyle\sum_{i \to j \in s} \left(\Psi(\lambda_{k,i \to j}) - \Psi(\sum_{j'} \lambda_{k,i \to j'})\right)\}; \end{aligned} \tag{7}$$

$$\gamma_{dk} = \alpha_k + \textstyle\sum_n \sum_{s \in \Omega^{-1}(w_{dn})} \phi_{dnkt}; \tag{8}$$

$$\begin{aligned} \lambda_{k,i \to j} &= \beta_{i \to j} \\ &\quad + \textstyle\sum_d \sum_n \sum_{s \in \Omega'(w_{dn})} \phi_{dnkt} \mathbb{I}\left[i \to j \in s\right]; \end{aligned} \tag{9}$$

where $\Omega'(w_{dn})$ is the set of all paths that lead to word w_{dn} in the tree, and t represents one particular path in this set. $\mathbb{I}\left[i \to j \in s\right]$ is the indicator of whether path s contains an edge from node i to j.

4.3 Hybrid Stochastic Inference

Given the complementary strengths of MCMC and VB, and following hybrid inference proposed by Mimno et al. (2012), we also derive hybrid inference for ptLDA.

The transition distributions $\boldsymbol{\pi}$ are treated identically as in variational inference. We posit a variational Dirichlet distribution $\boldsymbol{\lambda}$ and choose the one that minimizes the KL divergence between the true posterior and the variational distribution.

For topic $\boldsymbol{z}$ and path $\boldsymbol{y}$, instead of variational updates, we use a Gibbs sampler within a document. We sample z_{dn} and y_{dn} conditioned on the topic

and path assignments of all other document tokens, based on the variational expectation of $\boldsymbol{\pi}$,

$$q(z_{dn} = k, y_{dn} = s | \neg \boldsymbol{z}_{dn}, \neg \boldsymbol{y}_{dn}; \boldsymbol{w}) \propto \quad (10)$$
$$(\alpha + \textstyle\sum_{m \neq n} \mathbb{I}\left[z_{dm} = k\right])$$
$$\cdot \exp\{\mathbb{E}_q[\log p(y_{dn}|z_{dn}, \boldsymbol{\pi}) p(w_{dn}|y_{dn})]\}.$$

This equation embodies how this is a hybrid algorithm: the first term resembles the Gibbs sampling term encoding how much a document prefers a topic, while the second term encodes the expectation under the variational distribution of how much a path is preferred by this topic,

$$\mathbb{E}_q[\log p(y_{dn}|z_{dn}, \boldsymbol{\pi}) p(w_{dn}|y_{dn})] = \mathbb{I}_{[\Omega(y_{dn}) = w_{dn}]}$$
$$\cdot \textstyle\sum_{i \to j \in y_{dn}} \mathbb{E}_q[\log \lambda_{z_{dn}, i \to j}].$$

For every document, we sweep over all its tokens and resample their topic z_{dn} and path y_{dn} conditioned on all the other tokens' topic and path assignments $\neg \boldsymbol{z}_{dn}$ and $\neg \boldsymbol{y}_{dn}$. To avoid bias, we discard the first B burn-in sweeps and take the following M samples. We then use the empirical average of these samples update the global variational parameter $q(\boldsymbol{\pi}|\boldsymbol{\lambda})$ based on how many times we sampled these paths

$$\lambda_{k, i \to j} = \tfrac{1}{M} \textstyle\sum_d \sum_n \sum_{s \in \Omega^{-1}(w_{dn})} \big(\mathbb{I}\left[i \to j \in s\right]$$
$$\cdot \mathbb{I}\left[z_{dn} = k, y_{dn} = s\right]\big) + \beta_{i \to j}. \quad (11)$$

For our experiments, we use the recommended settings $B = 5$ and $M = 5$ from Mimno et al. (2012).

5 Experiments

We evaluate our new topic model, ptLDA, and existing topic models—LDA, pLDA, and tLDA—on their ability to induce domains for machine translation and the resulting performance of the translations on standard machine translation metrics.

Dataset and SMT Pipeline We use the NIST MT Chinese-English parallel corpus (NIST), excluding non-UN and non-HK Hansards portions as our training dataset. It contains 1.6M sentence pairs, with 40.4M Chinese tokens and 44.4M English tokens. We replicate the SMT pipeline of Eidelman et al. (2012): word segmentation (Tseng et al., 2005), align (Och and Ney, 2003), and symmetrize (Koehn et al., 2003) the data. We train a modified Kneser-Ney trigram language model on English (Chen and Goodman, 1996). We use CDEC (Dyer et al., 2010) for decoding, and MIRA (Crammer et al., 2006) for parameter training. To optimize SMT system, we tune the parameters on NIST MT06, and report results on three test sets: MT02, MT03 and MT05.[2]

Topic Models Configuration We compare our polylingual tree-based topic model (ptLDA) against tree-based topic models (tLDA), polylingual topic models (pLDA) and vanilla topic models (LDA).[3] We also examine different inference algorithms—Gibbs sampling (**gibbs**), variational inference (**variational**) and hybrid approach (**variational-hybrid**)—on the effects of SMT performance. In all experiments, we set the per-document Dirichlet parameter $\alpha = 0.01$ and the number of topics to 10, as used in Eidelman et al. (2012).

Resources for Prior Tree To build the tree for tLDA and ptLDA, we extract the word correlations from a Chinese-English bilingual dictionary (Denisowski, 1997).[4] We filter the dictionary using the NIST vocabulary, and keep entries mapping single Chinese and single English words. The prior tree has about 1000 word pairs (*dict*).

We also extract the bidirectional word alignments between Chinese and English using GIZA++ (Och and Ney, 2003). We then remove the word pairs appearing more than 50K times or fewer than 500 times and construct a second prior tree with about 2500 word pairs (*align*).

We apply both trees to tLDA and ptLDA, denoted as tLDA-*dict*, tLDA-*align*, ptLDA-*dict*, and ptLDA-*align*. However, tLDA-*align* and ptLDA-*align* do worse than tLDA-*dict* and ptLDA-*dict*, so we omit tLDA-*align* in the results.

Domain Adaptation using Topic Models We examine the effectiveness of using topic models for domain adaptation on standard SMT evaluation metrics—BLEU (Papineni et al., 2002) and TER (Snover et al., 2006). We report the results on three different test sets (Figure 2), and all SMT results are averaged over five runs.

We refer to the SMT model without domain adaptation as **baseline**.[5] LDA marginally improves machine translation (less than half a BLEU point).

[2] The NIST datasets contain 878, 919, 1082 and 1664 sentences for MT02, MT03, MT05 and MT06 respectively.

[3] For Gibbs sampling, we use implementations available in Hu and Boyd-Graber (2012) for tLDA; and Mallet (McCallum, 2002) for LDA and pLDA.

[4] This is a two-level tree structure. However, one could build a more sophisticated tree prior with a hierarchical dictionary such as multilingual WordNet.

[5] Our replication of Eidelman et al. (2012) yields slightly higher baseline performance, but the trend is consistent.

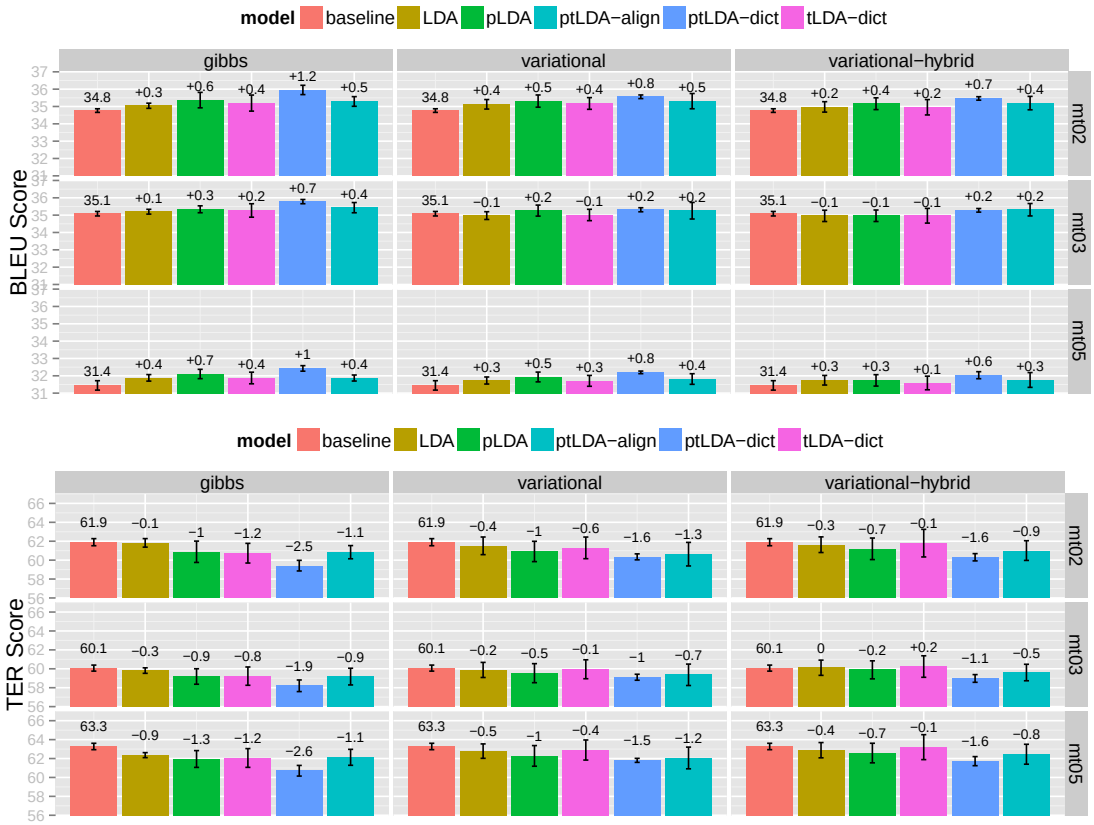


Figure 2: Machine translation performance for different models and inference algorithms against the baseline, on BLEU (top, higher the better) and TER (bottom, lower the better) scores. Our proposed ptLDA performs best. Results are averaged over 5 random runs. For model ptLDA-*dict* with different inference schemes, the BLEU improvement on three test sets is mostly significant with $p = 0.01$, except the results on MT03 using variational and variational-hybrid inferences.

Polylingual topic models pLDA and tree-based topic models tLDA-*dict* are consistently better than LDA, suggesting that incorporating additional bilingual knowledge improves topic models. These improvements are not redundant: our new ptLDA-*dict* model, which has aspects of both models yields the best performance among these approaches—up to a 1.2 BLEU point gain (higher is better), and -2.6 TER improvement (lower is better). The BLEU improvement is significant (Koehn, 2004) at $p = 0.01$,[6] except on MT03 with variational and variational-hybrid inference.

While ptLDA-*align* performs better than **baseline** SMT and LDA, it is worse than ptLDA-*dict*, possibly because of errors in the word alignments, making the tree priors less effective.

Scalability While **gibbs** has better translation scores than **variational** and **variational-hybrid**, it is less scalable to larger datasets. With 1.6M NIST training sentences, **gibbs** takes nearly a week to run 1000 iterations. In contrast, the parallelized **variational** and **variational-hybrid** approaches, which we implement in MapReduce (Dean and Ghemawat, 2004; Wolfe et al., 2008; Zhai et al., 2012), take less than a day to converge.

6 Discussion

In this section, we qualitatively analyze the translation results and investigate how ptLDA and its cousins improve SMT. We also discuss other approaches to improve unsupervised domain adaptation for SMT.

6.1 How do Topic Models Help SMT?

We present two examples of how topic models can improve SMT. The first example shows both LDA and ptLDA improve the **baseline**. The second example shows how LDA introduce biases that mislead SMT and how ptLDA's bilingual constraints correct these mistakes.

Figure 3 shows a sentence about a company

[6] Because we have multiple runs of each topic model (and thus different translation models), we select the run closest to the average BLEU for the translation significance test.

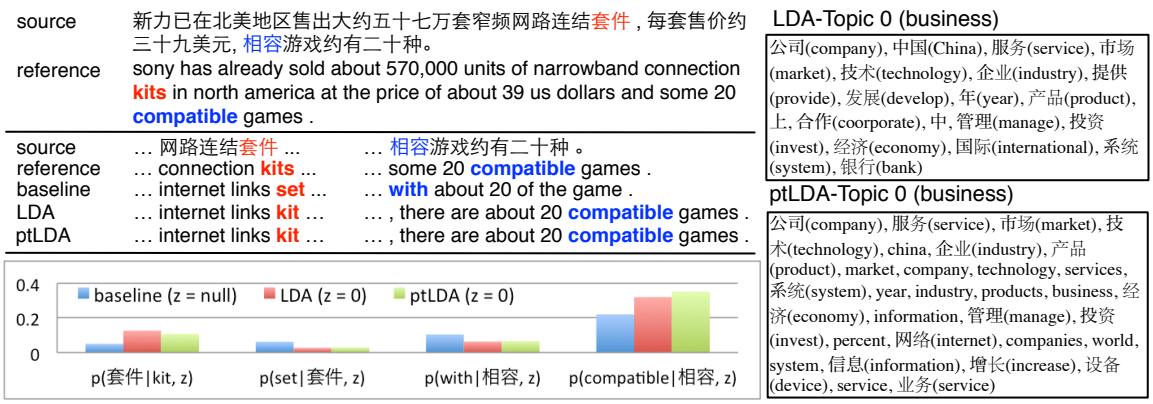


Figure 3: Better SMT result using topic models for domain adaptation. Top row: the source sentence and its reference translation. Middle row: the highlighted translations from different approaches. Bottom row: the change of relevant translation probabilities after incorporating the domain knowledge from LDA and ptLDA. Right: most-probable words of the topic the source sentence is assigned to under LDA (top) and ptLDA (bottom). The Chinese translations are in parenthesis.

introducing new technology gadgets where both LDA and ptLDA improve translations. The **baseline** translates "套件" to "set" (red), and "相容" to "with" (blue), which do not capture the reference meaning of a *add-on device* that works with *compatible* games. Both LDA and ptLDA assign this sentence to a business domain, which makes the translations probabilities shift toward correct translations: the probability of translating "相容" to "compatible" and the probability of translating "套件" to "kit" in the business domain are both significantly larger than without the domain knowledge; and the probabilities of translating "相容" to "with" and the probability of translating "set" to "套件" in the business domain decrease.

The second example (Figure 4) illustrates how ptLDA offers further improvements over LDA. The source sentence discusses foreign affairs. The **baseline** correctly translates the word "影响" to "affect". However, LDA—which only takes monolingual information from the source language—assigns this sentence to economic development. This misleads SMT to lower the probability for the correct translation "affect"; it chooses "impact" instead. In contrast, ptLDA—which incorporates bilingual constraints—successfully labels this sentence as foreign affairs and produces a softer, more nuanced translation that better matches the reference. The translation of "承诺" is very similar, except in this case, both the **baseline** and LDA produce the incorrect translation "the commitment of". This is possible because the probabilities of translating "承诺" to "promised to" and translating "promised to" to "承诺" (the correct translation, in both directions) increase when conditioned on ptLDA's correct topic but decrease when conditioned on LDA's incorrect topic.

6.2 Other Approaches

Other approaches have used topic models for machine translation. Xiao et al. (2012) present a topic similarity model based on LDA that produces a feature that weights grammar rules based on topic compatibility. They also model the source and target side of rules and compare the target similarity during decoding by projecting the target distribution into the source space. Hasler et al. (2012) use the source-side topic assignments from *hidden topic Markov models* (Gruber et al., 2007, HTMM) which models documents as a Markov chain and assign one topic to the whole sentence, instead of a mixture of topics. Su et al. (2012) also apply HTMM to monolingual data and apply the results to machine translation. To our knowledge, however, this is the first work to use *multilingual* topic models for domain adaptation in machine translation.

6.3 Improving Language Models

Topic models capture document-level properties of language, but a critical component of machine translation systems is the language model, which provides local constraints and preferences. Domain adaptation for language models (Bellegarda, 2004; Wood and Teh, 2009) is an important avenue for improving machine translation. Models that simultaneously discover global document themes as well as local, contextual domain-specific informa-

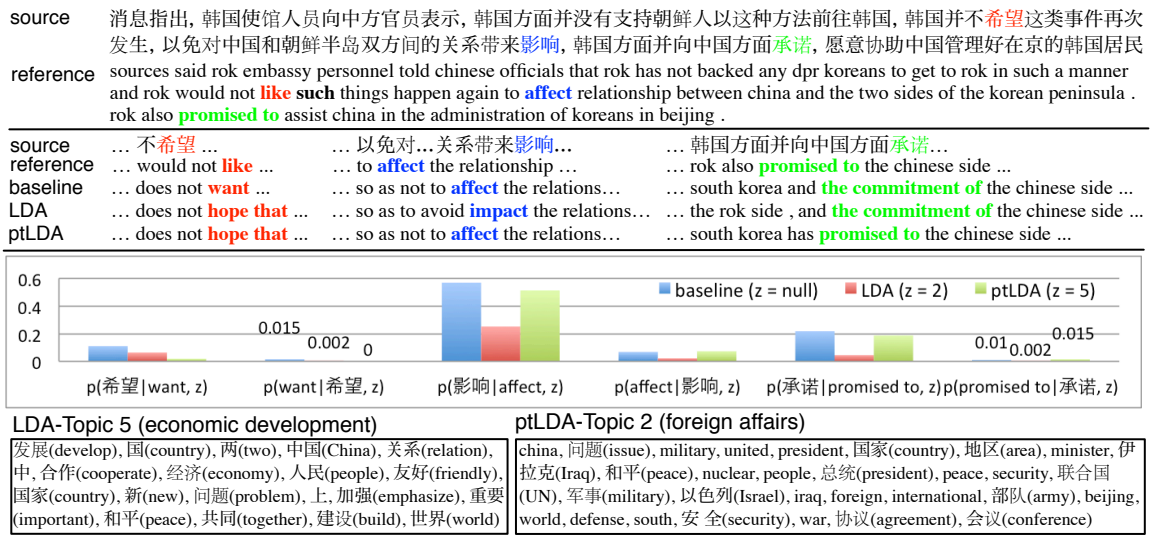


Figure 4: Better SMT result using ptLDA compared to LDA and the baseline. Top row: the source sentence and a reference translation. Second row: the highlighted translations from different models. Third row: the change of relevant translation probabilities after incorporating domain knowledge from LDA and ptLDA. Bottom row: most-probable words for the topics the source sentence is assigned to under LDA (left) and ptLDA (right). The meanings of Chinese words are in parenthesis.

tion (Wallach, 2006; Boyd-Graber and Blei, 2008) may offer further improvements.

6.4 External Data

The topic models presented here only require weak alignment between documents at the document level. Extending to larger datasets for learning topics is straightforward in principle. For example, ptLDA could learn domains from a much larger corpus like Wikipedia and then apply the extracted domains to machine translation data. However, this presents further challenges, as Wikipedia's domains are not representative of newswire machine translation datasets; a flexible hierarchical topic model (Teh et al., 2006) would better distinguish useful domains from extraneous ones.

7 Conclusion

Topic models generate great interest, but their use in "real world" applications still lags; this is particularly true for multilingual topic models. As topic models become more integrated in commonplace applications, their adoption, understanding, and robustness will improve.

This paper contributes to the deeper integration of topic models into critical applications by presenting a new multilingual topic model, ptLDA, comparing it with other multilingual topic models on a machine translation task, and showing that these topic models improve machine translation. ptLDA models both source and target data to induce domains from both dictionaries and alignments. Further improvement is possible by incorporating topic models deeper in the decoding process and adding domain knowledge to the language model.

Acknowledgments

We would like to thank the anonymous reviewers, Doug Oard, and John Morgan for their helpful comments, and thank Junhui Li and Ke Wu for insightful discussions. This work was supported by NSF Grant IIS-1320538. Boyd-Graber is also supported by NSF Grant CCF-1018625. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsor.

References

David Andrzejewski, Xiaojin Zhu, and Mark Craven. 2009. Incorporating domain knowledge into topic modeling via Dirichlet forest priors. In *Proceedings of the International Conference of Machine Learning*.

Arthur Asuncion, Max Welling, Padhraic Smyth, and Yee Whye Teh. 2009. On smoothing and inference for topic models. In *Proceedings of Uncertainty in Artificial Intelligence*.

Jerome R. Bellegarda. 2004. Statistical language model adaptation: review and perspectives. volume 42, pages 93–108.

David M. Blei and John D. Lafferty. 2009. Visualizing topics with Multi-Word expressions. *arXiv*.

David M. Blei, Andrew Ng, and Michael Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3.

Jordan Boyd-Graber and David M. Blei. 2008. Syntactic topic models. In *Proceedings of Advances in Neural Information Processing Systems*.

Jordan Boyd-Graber and David M. Blei. 2009. Multilingual topic models for unaligned text. In *Proceedings of Uncertainty in Artificial Intelligence*.

Jordan Boyd-Graber and Philip Resnik. 2010. Holistic sentiment analysis across languages: Multilingual supervised latent Dirichlet allocation. In *Proceedings of Emperical Methods in Natural Language Processing*.

Jordan Boyd-Graber, David M. Blei, and Xiaojin Zhu. 2007. A topic model for word sense disambiguation. In *Proceedings of Emperical Methods in Natural Language Processing*.

Jonathan Chang, Jordan Boyd-Graber, Chong Wang, Sean Gerrish, and David M. Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Proceedings of Advances in Neural Information Processing Systems*.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the Association for Computational Linguistics*.

David Chiang, Steve DeNeefe, and Michael Pust. 2011. Two easy improvements to lexical weighting. In *Proceedings of the Human Language Technology Conference*.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: Simplified data processing on large clusters. In *Symposium on Operating System Design and Implementation*.

Paul Denisowski. 1997. CEDICT. http://www.mdbg.net/chindict/.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Jonathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of ACL System Demonstrations*.

Vladimir Eidelman, Jordan Boyd-Graber, and Philip Resnik. 2012. Topic models for dynamic translation model adaptation. In *Proceedings of the Association for Computational Linguistics*.

George Foster and Roland Kuhn. 2007. Mixture-model adaptation for smt. In *Proceedings of the Second Workshop on Statistical Machine Translation*.

Amit Gruber, Michael Rosen-Zvi, and Yair Weiss. 2007. Hidden topic Markov models. In *Artificial Intelligence and Statistics*.

Eva Hasler, Barry Haddow, and Philipp Koehn. 2012. Sparse lexicalised features and topic adaptation for SMT. In *Proceedings of IWSLT*.

Yuening Hu and Jordan Boyd-Graber. 2012. Efficient tree-based topic modeling. In *Proceedings of the Association for Computational Linguistics*.

Yuening Hu, Jordan Boyd-Graber, Brianna Satinoff, and Alison Smith. 2013. Interactive topic modeling. *Machine Learning Journal*.

Saurabh S. Kataria, Krishnan S. Kumar, Rajeev R. Rastogi, Prithviraj Sen, and Srinivasan H. Sengamedu. 2011. Entity disambiguation with hierarchical topic models. In *Knowledge Discovery and Data Mining*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of Emperical Methods in Natural Language Processing*.

Philipp Koehn. 2009. *Statistical Machine Translation*. Cambridge University Press.

Li Fei-Fei and Pietro Perona. 2005. A Bayesian hierarchical model for learning natural scene categories. In *Computer Vision and Pattern Recognition*.

Spyros Matsoukas, Antti-Veikko I. Rosti, and Bing Zhang. 2009. Discriminative corpus weight estimation for machine translation. In *Proceedings of Emperical Methods in Natural Language Processing*.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://www.cs.umass.edu/ mccallum/mallet.

David Mimno, Hanna Wallach, Jason Naradowsky, David Smith, and Andrew McCallum. 2009. Polylingual topic models. In *Proceedings of Emperical Methods in Natural Language Processing*.

David Mimno, Matthew Hoffman, and David Blei. 2012. Sparse stochastic inference for latent Dirichlet allocation. In *Proceedings of the International Conference of Machine Learning*.

Radford M. Neal. 1993. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, University of Toronto.

Radford M. Neal. 2000. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265.

Franz Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. In *Computational Linguistics*, volume 29(21), pages 19–51.

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

Alessandro Perina, Pietro Lovato, Vittorio Murino, and Manuele Bicego. 2010. Biologically-aware latent Dirichlet allocation (balda) for the classification of expression microarray. In *Proceedings of the 5th IAPR international conference on Pattern recognition in bioinformatics*, PRIB'10.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*.

Jinsong Su, Hua Wu, Haifeng Wang, Yidong Chen, Xiaodong Shi, Huailin Dong, and Qun Liu. 2012. Translation model adaptation for statistical machine translation with monolingual topic information. In *Proceedings of the Association for Computational Linguistics*.

Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. 2006. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter. In *SIGHAN Workshop on Chinese Language Processing*.

Hanna M. Wallach. 2006. Topic modeling: Beyond bag-of-words. In *Proceedings of the International Conference of Machine Learning*.

Jason Wolfe, Aria Haghighi, and Dan Klein. 2008. Fully distributed EM for very large datasets. In *Proceedings of the International Conference of Machine Learning*, pages 1184–1191.

Frank Wood and Yee Whye Teh. 2009. A hierarchical nonparametric Bayesian approach to statistical language model domain adaptation. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 12.

Xinyan Xiao, Deyi Xiong, Min Zhang, Qun Liu, and Shouxun Lin. 2012. A topic similarity model for hierarchical phrase-based translation. In *Proceedings of the Association for Computational Linguistics*.

Limin Yao, David Mimno, and Andrew McCallum. 2009. Efficient methods for topic model inference on streaming document collections. In *Knowledge Discovery and Data Mining*.

Ke Zhai, Jordan Boyd-Graber, Nima Asadi, and Mohamad Alkhouja. 2012. Mr. LDA: A flexible large scale topic modeling package using variational inference in mapreduce. In *Proceedings of World Wide Web Conference*.

Bing Zhao and Eric P. Xing. 2006. BiTAM: Bilingual topic admixture models for word alignment. In *Proceedings of the Association for Computational Linguistics*.

Low-Resource Semantic Role Labeling

Matthew R. Gormley[1] **Margaret Mitchell**[2] **Benjamin Van Durme**[1] **Mark Dredze**[1]
[1]Human Language Technology Center of Excellence
Johns Hopkins University, Baltimore, MD 21211
[2]Microsoft Research
Redmond, WA 98052
mrg@cs.jhu.edu | memitc@microsoft.com | vandurme@cs.jhu.edu | mdredze@cs.jhu.edu

Abstract

We explore the extent to which high-resource manual annotations such as treebanks are necessary for the task of semantic role labeling (SRL). We examine how performance changes without syntactic supervision, comparing both *joint* and *pipelined* methods to induce latent syntax. This work highlights a new application of unsupervised grammar induction and demonstrates several approaches to SRL in the absence of supervised syntax. Our best models obtain competitive results in the high-resource setting and state-of-the-art results in the low resource setting, reaching 72.48% F1 averaged across languages. We release our code for this work along with a larger toolkit for specifying arbitrary graphical structure.[1]

1 Introduction

The goal of semantic role labeling (SRL) is to identify predicates and arguments and label their semantic contribution in a sentence. Such labeling defines *who* did *what* to *whom*, *when*, *where* and *how*. For example, in the sentence "The kids ran the marathon", *ran* assigns a role to *kids* to denote that they are the runners; and a role to *marathon* to denote that it is the race course.

Models for SRL have increasingly come to rely on an array of NLP tools (e.g., parsers, lemmatizers) in order to obtain state-of-the-art results (Björkelund et al., 2009; Zhao et al., 2009). Each tool is typically trained on hand-annotated data, thus placing SRL at the end of a very high-resource NLP pipeline. However, richly annotated data such as that provided in parsing treebanks is expensive to produce, and may be tied to specific domains (e.g., newswire). Many languages do not have such supervised resources (*low-resource languages*), which makes exploring SRL cross-linguistically difficult.

The problem of SRL for low-resource languages is an important one to solve, as solutions pave the way for a wide range of applications: Accurate identification of the semantic roles of entities is a critical step for any application sensitive to semantics, from information retrieval to machine translation to question answering.

In this work, we explore models that minimize the need for high-resource supervision. We examine approaches in a **joint** setting where we marginalize over latent syntax to find the optimal semantic role assignment; and a **pipeline** setting where we first induce an unsupervised grammar. We find that the joint approach is a viable alternative for making reasonable semantic role predictions, outperforming the pipeline models. These models can be effectively trained with access to only SRL annotations, and mark a state-of-the-art contribution for low-resource SRL.

To better understand the effect of the low-resource grammars and features used in these models, we further include comparisons with (1) models that use higher-resource versions of the same features; (2) state-of-the-art high resource models; and (3) previous work on low-resource grammar induction. In sum, this paper makes several experimental and modeling contributions, summarized below.

Experimental contributions:

- Comparison of pipeline and joint models for SRL.
- Subtractive experiments that consider the removal of supervised data.
- Analysis of the induced grammars in unsupervised, distantly-supervised, and joint training settings.

[1]http://www.cs.jhu.edu/~mrg/software/

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1177–1187,
Baltimore, Maryland, USA, June 23-25 2014.

Modeling contributions:

- Simpler joint CRF for syntactic and semantic dependency parsing than previously reported.
- New application of unsupervised grammar induction: low-resource SRL.
- Constrained grammar induction using SRL for distant-supervision.
- Use of Brown clusters in place of POS tags for low-resource SRL.

The pipeline models are introduced in § 3.1 and jointly-trained models for syntactic and semantic dependencies (similar in form to Naradowsky et al. (2012)) are introduced in § 3.2. In the pipeline models, we develop a novel approach to unsupervised grammar induction and explore performance using SRL as distant supervision. The joint models use a non-loopy conditional random field (CRF) with a global factor constraining latent syntactic edge variables to form a tree. Efficient exact marginal inference is possible by embedding a dynamic programming algorithm within belief propagation as in Smith and Eisner (2008).

Even at the expense of no dependency path features, the joint models best pipeline-trained models for state-of-the-art performance in the low-resource setting (§ 4.4). When the models have access to observed syntactic trees, they achieve near state-of-the-art accuracy in the high-resource setting on some languages (§ 4.3).

Examining the learning curve of the joint and pipeline models in two languages demonstrates that a small number of labeled SRL examples may be essential for good end-task performance, but that the choice of a good model for grammar induction has an even greater impact.

2 Related Work

Our work builds upon research in both semantic role labeling and unsupervised grammar induction (Klein and Manning, 2004; Spitkovsky et al., 2010a). Previous related approaches to semantic role labeling include joint classification of semantic arguments (Toutanova et al., 2005; Johansson and Nugues, 2008), latent syntax induction (Boxwell et al., 2011; Naradowsky et al., 2012), and feature engineering for SRL (Zhao et al., 2009; Björkelund et al., 2009).

Toutanova et al. (2005) introduced one of the first joint approaches for SRL and demonstrated that a model that scores the full predicate-argument structure of a parse tree could lead to significant error reduction over independent classifiers for each predicate-argument relation.

Johansson and Nugues (2008) and Lluís et al. (2013) extend this idea by coupling predictions of a dependency parser with predictions from a semantic role labeler. In the model from Johansson and Nugues (2008), the outputs from an SRL pipeline are reranked based on the full predicate-argument structure that they form. The candidate set of syntactic-semantic structures is reranked using the probability of the syntactic tree and semantic structure. Lluís et al. (2013) use a joint arc-factored model that predicts full syntactic paths along with predicate-argument structures via dual decomposition.

Boxwell et al. (2011) and Naradowsky et al. (2012) observe that syntax may be treated as latent when a treebank is not available. Boxwell et al. (2011) describe a method for training a semantic role labeler by extracting features from a packed CCG parse chart, where the parse weights are given by a simple ruleset. Naradowsky et al. (2012) marginalize over latent syntactic dependency parses.

Both Boxwell et al. (2011) and Naradowsky et al. (2012) suggest methods for SRL without supervised syntax, however, their features come largely from supervised resources. Even in their lowest resource setting, Boxwell et al. (2011) require an oracle CCG tag dictionary extracted from a treebank. Naradowsky et al. (2012) limit their exploration to a small set of basic features, and included high-resource supervision in the form of lemmas, POS tags, and morphology available from the CoNLL 2009 data.

There has not yet been a comparison of techniques for SRL that do not rely on a syntactic treebank, and no exploration of probabilistic models for unsupervised grammar induction within an SRL pipeline that we have been able to find.

Related work for the unsupervised learning of dependency structures separately from semantic roles primarily comes from Klein and Manning (2004), who introduced the Dependency Model with Valence (DMV). This is a robust generative model that uses a head-outward process over word classes, where heads generate arguments.

Spitkovsky et al. (2010a) show that Viterbi (hard) EM training of the DMV with simple uniform initialization of the model parameters yields higher accuracy models than standard soft-EM

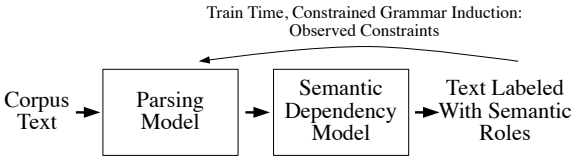


Figure 1: Pipeline approach to SRL. In this simple pipeline, the first stage syntactically parses the corpus, and the second stage predicts semantic predicate-argument structure for each sentence using the labels of the first stage as features. In our *low-resource* pipelines, we assume that the syntactic parser is given no labeled parses—however, it may optionally utilize the semantic parses as distant supervision. Our experiments also consider 'longer' pipelines that include earlier stages: a morphological analyzer, POS tagger, lemmatizer.

training. In Viterbi EM, the E-step finds the maximum likelihood corpus parse given the current model parameters. The M-step then finds the maximum likelihood parameters given the corpus parse. We utilize this approach to produce unsupervised syntactic features for the SRL task.

Grammar induction work has further demonstrated that distant supervision in the form of ACE-style relations (Naseem and Barzilay, 2011) or HTML markup (Spitkovsky et al., 2010b) can lead to considerable gains. Recent work in fully unsupervised dependency parsing has supplanted these methods with even higher accuracies (Spitkovsky et al., 2013) by arranging optimizers into networks that suggest informed restarts based on previously identified local optima. We do not reimplement these approaches within the SRL pipeline here, but provide comparison of these methods against our grammar induction approach in isolation in § 4.5.

In both pipeline and joint models, we use features adapted from state-of-the-art approaches to SRL. This includes Zhao et al. (2009) features, who use feature templates from combinations of word properties, syntactic positions including head and children, and semantic properties; and features from Björkelund et al. (2009), who utilize features on syntactic siblings and the dependency path concatenated with the direction of each edge. Features are described further in § 3.3.

3 Approaches

We consider an array of models, varying:

1. Pipeline vs. joint training (Figures 1 and 2)
2. Types of supervision
3. The objective function at the level of syntax

3.1 Unsupervised Syntax in the Pipeline

Typical SRL systems are trained following a pipeline where the first component is trained on supervised data, and each subsequent component is trained using the 1-best output of the previous components. A typical pipeline consists of a POS tagger, dependency parser, and semantic role labeler. In this section, we introduce pipelines that remove the need for a supervised tagger and parser by training in an unsupervised and distantly supervised fashion.

Brown Clusters We use fully unsupervised Brown clusters (Brown et al., 1992) in place of POS tags. Brown clusters have been used to good effect for various NLP tasks such as named entity recognition (Miller et al., 2004) and dependency parsing (Koo et al., 2008; Spitkovsky et al., 2011).

The clusters are formed by a greedy hierachical clustering algorithm that finds an assignment of words to classes by maximizing the likelihood of the training data under a latent-class bigram model. Each word type is assigned to a fine-grained cluster at a leaf of the hierarchy of clusters. Each cluster can be uniquely identified by the path from the root cluster to that leaf. Representing this path as a bit-string (with 1 indicating a left and 0 indicating a right child) allows a simple coarsening of the clusters by truncating the bit-strings. We train 1000 Brown clusters for each of the CoNLL-2009 languages on Wikipedia text.[2]

Unsupervised Grammar Induction Our first method for grammar induction is *fully unsupervised* Viterbi EM training of the Dependency Model with Valence (DMV) (Klein and Manning, 2004), with uniform initialization of the model parameters. We define the DMV such that it generates sequences of word classes: either POS tags or Brown clusters as in Spitkovsky et al. (2011). The DMV is a simple generative model for projective dependency trees. Children are generated recursively for each node. Conditioned on the parent class, the direction (right or left), and the current valence (first child or not), a coin is flipped to decide whether to generate another child; the distribution over child classes is conditioned on only the parent class and direction.

[2]The Wikipedia text was tokenized for Polyglot (Al-Rfou' et al., 2013): `http://bit.ly/embeddings`

Constrained Grammar Induction Our second method, which we will refer to as DMV+C, induces grammar in a *distantly supervised* fashion by using a constrained parser in the E-step of Viterbi EM. Since the parser is part of a pipeline, we constrain it to respect the downstream SRL annotations during training. At test time, the parser is unconstrained.

Dependency-based semantic role labeling can be described as a simple structured prediction problem: the predicted structure is a labeled directed graph, where nodes correspond to words in the sentence. Each directed edge indicates that there is a predicate-argument relationship between the two words; the parent is the predicate and the child the argument. The label on the edge indicates the type of semantic relationship. Unlike syntactic dependency parsing, the graph is not required to be a tree, nor even a connected graph. Self-loops and crossing arcs are permitted.

The constrained *syntactic* DMV parser treats the semantic graph as observed, and constrains the syntactic parent to be chosen from one of the semantic parents, if there are any. In some cases, imposing this constraint would not permit *any* projective dependency parses—in this case, we ignore the semantic constraint for that sentence. We parse with the CKY algorithm (Younger, 1967; Aho and Ullman, 1972) by utilizing a PCFG corresponding to the DMV (Cohn et al., 2010). Each chart cell allows only non-terminals compatible with the constrained sets. This can be viewed as a variation of Pereira and Schabes (1992).

Semantic Dependency Model As described above, semantic role labeling can be cast as a structured prediction problem where the structure is a labeled semantic dependency graph. We define a conditional random field (CRF) (Lafferty et al., 2001) for this task. Because each word in a sentence may be in a semantic relationship with any other word (including itself), a sentence of length n has n^2 possible edges. We define a single L+1-ary variable for each edge, whose value can be any of L semantic labels or a special label indicating there is no predicate-argument relationship between the two words. In this way, we jointly perform *identification* (determining whether a semantic relationship exists) and *classification* (determining the semantic label). This use of an L+1-ary variable is in contrast to the model of Naradowsky et al. (2012), which used a more complex set of binary variables and required a constraint factor permitting AT-MOST-ONE. We include one unary factor for each variable.

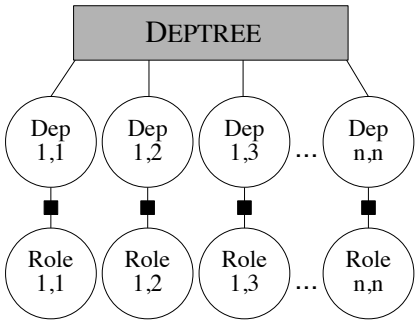


Figure 2: Factor graph for the joint syntactic/semantic dependency parsing model.

We optionally include additional variables that perform word sense disambiguation for each predicate. Each has a unary factor and is completely disconnected from the semantic edge (similar to Naradowsky et al. (2012)). These variables range over all the predicate senses observed in the training data for the *lemma* of that predicate.

3.2 Joint Syntactic and Semantic Parsing Model

In Section 3.1, we introduced pipeline-trained models for SRL, which used grammar induction to predict unlabeled syntactic parses. In this section, we define a simple model for joint syntactic and semantic dependency parsing.

This model extends the CRF model in Section 3.1 to include the projective syntactic dependency parse for a sentence. This is done by including an additional n^2 binary variables that indicate whether or not a directed syntactic dependency edge exists between a pair of words in the sentence. Unlike the semantic dependencies, these syntactic variables must be coupled so that they produce a projective dependency parse; this requires an additional global constraint factor to ensure that this is the case (Smith and Eisner, 2008). The constraint factor touches all n^2 syntactic-edge variables, and multiplies in 1.0 if they form a projective dependency parse, and 0.0 otherwise. We couple each syntactic edge variable to its semantic edge variable with a binary factor. Figure 2 shows the factor graph for this joint model.

Note that our factor graph does not contain any loops, thereby permitting efficient exact marginal inference just as in Naradowsky et al. (2012). We

Property		Possible values
1	word form	all word forms
2	lower case word form	all lower-case forms
3	5-char word form prefixes	all 5-char form prefixes
4	capitalization	*True, False*
5	top-800 word form	top-800 word forms
6	brown cluster	*000, 1100, 010110001, ...*
7	brown cluster, length 5	length 5 prefixes of brown clusters
8	lemma	all word lemmas
9	POS tag	*NNP, CD, JJ, DT, ...*
10	morphological features (different across languages)	Gender, Case, Number, ...
11	dependency label	*SBJ, NMOD, LOC, ...*
12	edge direction	*Up, Down*

Table 1: Word and edge properties in templates.

i, i-1, i+1	noFarChildren(w_i)	linePath(w_p, w_c)
parent(w_i)	rightNearSib(w_i)	depPath(w_p, w_c)
allChildren(w_i)	leftNearSib(w_i)	depPath(w_p, w_{lca})
rightNearChild(w_i)	firstVSupp(w_i)	depPath(w_c, w_{lca})
rightFarChild(w_i)	lastVSupp(w_i)	depPath(w_{lca}, w_{root})
leftNearChild(w_i)	firstNSupp(w_i)	
leftFarChild(w_i)	lastNSupp(w_i)	

Table 2: Word positions used in templates. Based on current word position (i), positions related to current word w_i, possible parent, child (w_p, w_c), lowest common ancestor between parent/child (w_{lca}), and syntactic root (w_{root}).

train our CRF models by maximizing conditional log-likelihood using stochastic gradient descent with an adaptive learning rate (AdaGrad) (Duchi et al., 2011) over mini-batches.

The unary and binary factors are defined with exponential family potentials. In the next section, we consider binary features of the observations (the sentence and labels from previous pipeline stages) which are conjoined with the state of the variables in the factor.

3.3 Features for CRF Models

Our feature design stems from two key ideas. First, for SRL, it has been observed that feature bigrams (the concatenation of simple features such as a predicate's POS tag and an argument's word) are important for state-of-the-art (Zhao et al., 2009; Björkelund et al., 2009). Second, for syntactic dependency parsing, combining Brown cluster features with word forms or POS tags yields high accuracy even with little training data (Koo et al., 2008).

We create binary indicator features for each model using feature templates. Our feature template definitions build from those used by the top performing systems in the CoNLL-2009 Shared Task, Zhao et al. (2009) and Björkelund et al. (2009) and from features in syntactic dependency parsing (McDonald et al., 2005; Koo et al., 2008).

Template	Possible values
relative position	*before, after, on*
distance, continuity	$\mathbb{Z}^+$
binned distance	$>$ 2, 5, 10, 20, 30, or 40
geneological relationship	*parent, child, ancestor, descendant*
path-grams	*the_NN_went*

Table 3: Additional standalone templates.

Template Creation Feature templates are defined over triples of $\langle$property, positions, order$\rangle$. **Properties**, listed in Table 1, are extracted from word **positions** within the sentence, shown in Table 2. Single positions for a word w_i include its syntactic parent, its leftmost farthest child (leftFarChild), its rightmost nearest sibling (rightNearSib), etc. Following Zhao et al. (2009), we include the notion of verb and noun supports and sections of the dependency path. Also following Zhao et al. (2009), properties from a set of positions can be put together in three possible **orders**: as the given sequence, as a sorted list of unique strings, and removing all duplicated neighbored strings. We consider both template unigrams and bigrams, combining two templates in sequence.

Additional templates we include are the relative position (Björkelund et al., 2009), geneological relationship, distance (Zhao et al., 2009), and binned distance (Koo et al., 2008) between two words in the path. From Lluís et al. (2013), we use $1, 2, 3$-gram path features of words/POS tags (path-grams), and the number of non-consecutive token pairs in a predicate-argument path (continuity).

3.4 Feature Selection

Constructing all feature template unigrams and bigrams would yield an unwieldy number of features. We therefore determine the top N template bigrams for a dataset and factor a according to an information gain measure (Martins et al., 2011):

$$IG_{a,m} = \sum_{f \in T_m} \sum_{x_a} p(f, x_a) \log_2 \frac{p(f, x_a)}{p(f)p(x_a)}$$

where T_m is the mth feature template, f is a particular instantiation of that template, and x_a is an assignment to the variables in factor a. The probabilities are empirical estimates computed from the training data. This is simply the mutual information of the feature template instantiation with the variable assignment.

This filtering approach was treated as a simple baseline in Martins et al. (2011) to contrast with increasingly popular gradient based regularization approaches. Unlike the gradient based ap-

proaches, this filtering approach easily scales to many features since we can decompose the memory usage over feature templates.

As an additional speedup, we reduce the dimensionality of our feature space to 1 million for each clique using a common trick referred to as *feature hashing* (Weinberger et al., 2009): we map each feature instantiation to an integer using a hash function[3] modulo the desired dimentionality.

4 Experiments

We are interested in the effects of varied supervision using pipeline and joint training for SRL. To compare to prior work (i.e., submissions to the CoNLL-2009 Shared Task), we also consider the joint task of semantic role labeling *and* predicate sense disambiguation. Our experiments are subtractive, beginning with all supervision available and then successively removing (a) dependency syntax, (b) morphological features, (c) POS tags, and (d) lemmas. Dependency syntax is the most expensive and difficult to obtain of these various forms of supervision. We explore the importance of both the labels and structure, and what quantity of supervision is useful.

4.1 Data

The CoNLL-2009 Shared Task (Hajič et al., 2009) dataset contains POS tags, lemmas, morphological features, syntactic dependencies, predicate senses, and semantic roles annotations for 7 languages: Catalan, Chinese, Czech, English, German, Japanese,[4] Spanish. The CoNLL-2005 and -2008 Shared Task datasets provide English SRL annotation, and for cross dataset comparability we consider only verbal predicates (more details in § 4.4). To compare with prior approaches that use semantic supervision for grammar induction, we utilize Section 23 of the WSJ portion of the Penn Treebank (Marcus et al., 1993).

4.2 Feature Template Sets

Our primary feature set $\mathbf{IG}_C$ consists of 127 template unigrams that emphasize coarse properties (i.e., properties 7, 9, and 11 in Table 1). We also explore the 31 template unigrams[5] $\mathbf{IG}_B$ described by Björkelund et al. (2009). Each of IG_C and IG_B also include 32 template bigrams selected by information gain on 1000 sentences—we select a different set of template bigrams for each dataset.

We compare against the language-specific feature sets detailed in the literature on high-resource top-performing SRL systems: From Björkelund et al. (2009), these are feature sets for German, English, Spanish and Chinese, obtained by weeks of forward selection ($\mathbf{B}_{de,en,es,zh}$); and from Zhao et al. (2009), these are features for Catalan $\mathbf{Z}_{ca}$.[6]

4.3 High-resource SRL

We first compare our models trained as a pipeline, using all available supervision (syntax, morphology, POS tags, lemmas) from the CoNLL-2009 data. Table 4(a) shows the results of our model with gold syntax and a richer feature set than that of Naradowsky et al. (2012), which only looked at whether a syntactic dependency edge was present. This highlights an important advantage of the pipeline trained model: the features can consider any part of the syntax (e.g., arbitrary subtrees), whereas the joint model is limited to those features over which it can efficiently marginalize (e.g., short dependency paths). This holds true even in the pipeline setting where no syntactic supervision is available.

Table 4(b) contrasts our high-resource results for the task of SRL and sense disambiguation with the top systems in the CoNLL-2009 Shared Task, giving further insight into the performance of the simple information gain feature selection technique. With supervised syntax, our simple information gain feature selection technique (§ 3.4) performs admirably. However, the original unigram Björkelund features ($\text{B}_{de,en,es,zh}$), which were tuned for a high-resource model, obtain higher F1 than our information gain set using the same features in unigram and bigram templates (IG_B). This suggests that further work on feature selection may improve the results. We find that IG_B obtain *higher* F1 than the original Björkelund feature sets ($\text{B}_{de,en,es,zh}$) in the low-resource pipeline setting with constrained grammar induction (DMV+C).

[3] To reduce hash collisions, We use MurmurHash v3 `https://code.google.com/p/smhasher`.

[4] We do not report results on Japanese as that data was only made freely available to researchers that competed in CoNLL 2009.

[5] Because we do not include a binary factor between predicate sense and semantic role, we do not include sense as a feature for argument prediction.

[6] This covers all CoNLL languages but Czech, where feature sets were not made publicly available in either work. In Czech, we disallowed template bigrams involving path-grams.

	SRL Approach	Feature Set	Dep. Parser	Avg.	ca	cs	de	en	es	zh
(a)	Pipeline	IG_C	Gold	**84.98**	84.97	**87.65**	79.14	**86.54**	84.22	**87.35**
	Pipeline	IG_B	Gold	84.74	**85.15**	86.64	**79.50**	85.77	**84.40**	86.95
	Naradowsky et al. (2012)		Gold	72.73	69.59	74.84	66.49	78.55	68.93	77.97
(b)	Björkelund et al. (2009)		Supervised	**81.55**	80.01	**85.41**	**79.71**	**85.63**	79.91	**78.60**
	Zhao et al. (2009)		Supervised	80.85	**80.32**	85.19	75.99	85.44	**80.46**	77.72
	Pipeline	IG_C	Supervised	78.03	76.24	83.34	74.19	81.96	76.12	76.35
	Pipeline	Z_{ca}	Supervised	*77.62	77.62	—	—	—	—	—
	Pipeline	$B_{de,en,es,zh}$	Supervised	*76.49	—	—	72.17	81.15	76.65	75.99
	Pipeline	IG_B	Supervised	75.68	74.59	81.61	69.08	78.86	74.51	75.44
(c)	Joint	IG_C	Marginalized	**72.48**	71.35	**81.03**	65.15	**76.16**	71.03	70.14
	Joint	IG_B	Marginalized	72.40	**71.55**	80.04	64.80	75.57	**71.21**	71.21
	Naradowsky et al. (2012)		Marginalized	71.27	67.99	73.16	**67.26**	76.12	66.74	**76.32**
	Pipeline	IG_C	DMV+C (bc)	70.08	68.21	79.63	62.25	73.81	68.73	67.86
	Pipeline	Z_{ca}	DMV+C (bc)	*69.67	69.67	—	—	—	—	—
	Pipeline	IG_C	DMV (bc)	69.26	68.04	79.58	58.47	74.78	68.36	66.35
	Pipeline	IG_B	DMV (bc)	66.81	63.31	77.38	59.91	72.02	65.96	62.28
	Pipeline	IG_B	DMV+C (bc)	65.61	61.89	77.48	58.97	69.11	63.31	62.92
	Pipeline	$B_{de,en,es,zh}$	DMV+C (bc)	*63.06	—	—	57.75	68.32	63.70	62.45

Table 4: Test F1 for SRL and sense disambiguation on CoNLL'09 in high-resource and low-resource settings: we study (a) gold syntax, (b) supervised syntax, and (c) unsupervised syntax. Results are ranked by F1 with bold numbers indicating the best F1 for a language and level of supervision.
*Indicates partial averages for the language-specific feature sets (Z_{ca} and $B_{de,en,es,zh}$), for which we show results only on the languages for which the sets were publicly available.

train \ test		*2008 heads*	*2005 spans*	*2005 spans (oracle tree)*
☑ PRY'08	*2005 spans*	84.32	79.44	
☐ B'11 (tdc)		—	71.5	
☐ B'11 (td)		—	65.0	
☑ JN'08	*2008 heads*	85.93	79.90	
☐ Joint, IG_C		72.9	35.0	72.0
☐ Joint, IG_B		67.3	37.8	67.1

Table 5: F1 for SRL approaches (without sense disambiguation) in matched and mismatched train/test settings for CoNLL 2005 span and 2008 head supervision. We contrast low-resource (☐) and high-resource settings (☑), where latter uses a treebank. See § 4.4 for caveats to this comparison.

4.4 Low-Resource SRL

CoNLL-2009 Table 4(c) includes results for our low-resource approaches and Naradowsky et al. (2012) on predicting semantic roles as well as sense. In the low-resource setting of the CoNLL-2009 Shared task without syntactic supervision, our joint model (Joint) with marginalized syntax obtains state-of-the-art results with features IG_C described in § 4.2. This model outperforms prior work (Naradowsky et al., 2012) and our pipeline model (Pipeline) with contrained (DMV+C) and unconstrained grammar induction (DMV) trained on brown clusters (bc).

In the low-resource setting, training and decoding times for the pipeline and joint methods are similar as computation time tends to be dominated by feature extraction.

These results begin to answer a key research question in this work: The joint models outperform the pipeline models in the low-resource setting. This holds even when using the same feature selection process. Further, the best-performing low-resource features found in this work are those based on coarse feature templates and selected by information gain. Templates for these features generalize well to the high-resource setting. However, analysis of the induced grammars in the pipeline setting suggests that the book is not closed on the issue. We return to this in § 4.5.

CoNLL-2008, -2005 To finish out comparisons with state-of-the-art SRL, we contrast our approach with that of Boxwell et al. (2011), who evaluate on SRL in isolation (without sense disambiguation, as in CoNLL-2009). They report results on Prop-CCGbank (Boxwell and White, 2008), which uses the same training/testing splits as the CoNLL-2005 Shared Task. Their results are therefore loosely[7] comparable to results on the CoNLL-2005 dataset, which we can compare here.

There is an additional complication in comparing SRL approaches directly: The CoNLL-2005 dataset defines arguments as *spans* instead of

[7]The comparison is imperfect for two reasons: first, the CCGBank contains only 99.44% of the original PTB sentences (Hockenmaier and Steedman, 2007); second, because PropBank was annotated over CFGs, after converting to CCG only 99.977% of the argument spans were exact matches (Boxwell and White, 2008). However, this comparison was adopted by Boxwell et al. (2011), so we use it here.

heads, which runs counter to our head-based syntactic representation. This creates a mismatched train/test scenario: we must train our model to predict argument *heads*, but then test on our models ability to predict argument *spans*.[8] We therefore train our models on the CoNLL-2008 argument heads,[9] and post-process and convert from heads to spans using the conversion algorithm available from Johansson and Nugues (2008).[10] The heads are either from an MBR tree or an oracle tree. This gives Boxwell et al. (2011) the advantage, since our syntactic dependency parses are optimized to pick out semantic argument heads, not spans.

Table 5 presents our results. Boxwell et al. (2011) (B'11) uses additional supervision in the form of a CCG tag dictionary derived from supervised data with (tdc) and without (tc) a cutoff. Our model does very poorly on the '05 span-based evaluation because the constituent bracketing of the marginalized trees are inaccurate. This is elucidated by instead evaluating on the oracle spans, where our F1 scores are higher than Boxwell et al. (2011). We also contrast with relavant high-resource methods with span/head conversions from Johansson and Nugues (2008): Punyakanok et al. (2008) (PRY'08) and Johansson and Nugues (2008) (JN'08).

Subtractive Study In our subsequent experiments, we study the effectiveness of our models as the available supervision is decreased. We incrementally remove dependency syntax, morphological features, POS tags, then lemmas. For these experiments, we utilize the coarse-grained feature set (IG_C), which includes Brown clusters.

Across languages, we find the largest drop in F1 when we remove POS tags; and we find a gain in F1 when we remove lemmas. This indicates that lemmas, which are a high-resource annotation, may not provide a significant benefit for this task. The effect of removing morphological features is different across languages, with little change in performance for Catalan and Spanish, but a drop in performance for German. This may reflect a difference between the languages, or may reflect the difference between the annotation of the languages: both the Catalan and Spanish data originated from the Ancora project,[11] while the German data came from another source.

Figure 3 contains the learning curve for SRL supervision in our lowest resource setting for two example languages, Catalan and German. This shows how F1 of SRL changes as we adjust the number of training examples. We find that the joint training approach to grammar induction yields consistently higher SRL performance than its distantly supervised counterpart.

Rem	#FT	ca	de	es
–	127+32	74.46	72.62	74.23
Dep	40+32	67.43	64.24	67.18
Mor	30+32	67.84	59.78	66.94
POS	23+32	64.40	54.68	62.71
Lem	21+32	64.85	54.89	63.80

Table 6: Subtractive experiments. Each row contains the F1 for SRL only (without sense disambiguation) where the supervision type of that row and all above it have been removed. Removed supervision types (Rem) are: syntactic dependencies (*Dep*), morphology (*Mor*), POS tags (*POS*), and lemmas (*Lem*). #FT indicates the number of feature templates used (unigrams+bigrams).

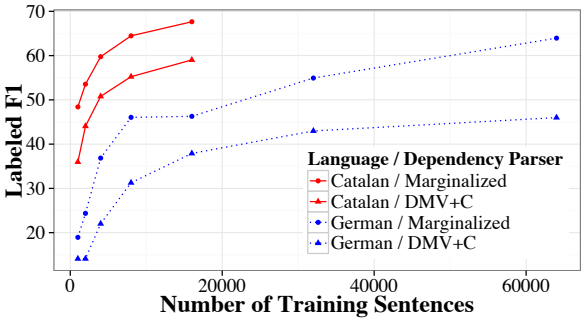


Figure 3: Learning curve for semantic dependency supervision in Catalan and German. F1 of SRL only (without sense disambiguation) shown as the number of training sentences is increased.

4.5 Analysis of Grammar Induction

Table 7 shows grammar induction accuracy in low-resource settings. We find that the gap between the supervised parser and the unsupervised methods is quite large, despite the reasonable accuracy both methods achieve for the SRL end task.

[8] We were unable to obtain the system output of Boxwell et al. (2011) in order to convert their spans to dependencies and evaluate the other mismatched train/test setting.

[9] CoNLL-2005, -2008, and -2009 were derived from PropBank and share the same source text; -2008 and -2009 use argument heads.

[10] Specifically, we use their Algorithm 2, which produces the span dominated by each argument, with special handling of the case when the argument head dominates that of the predicate. Also following Johansson and Nugues (2008), we recover the '05 sentences missing from the '08 evaluation set.

[11] http://clic.ub.edu/corpus/ancora

Dependency Parser	Avg.	ca	cs	de	en	es	zh
Supervised*	87.1	89.4	85.3	89.6	88.4	89.2	80.7
DMV (pos)	30.2	45.3	22.7	20.9	32.9	41.9	17.2
DMV (bc)	22.1	18.8	32.8	19.6	22.4	20.5	18.6
DMV+C (pos)	37.5	50.2	34.9	21.5	36.9	49.8	32.0
DMV+C (bc)	40.2	46.3	37.5	28.7	40.6	50.4	37.5
Marginal, IG_C	43.8	50.3	45.8	27.2	44.2	46.3	48.5
Marginal, IG_B	50.2	52.4	43.4	41.3	52.6	55.2	56.2

Table 7: Unlabeled directed dependency accuracy on CoNLL'09 test set in low-resource settings. DMV models are trained on either POS tags (pos) or Brown clusters (bc). *Indicates the supervised parser outputs provided by the CoNLL'09 Shared Task.

	WSJ^{∞}	Distant Supervision
SAJM'10	44.8	none
SAJ'13	**64.4**	none
SJA'10	50.4	HTML
NB'11	59.4	ACE05
DMV (bc)	24.8	none
DMV+C (bc)	44.8	SRL
Marginalized, IG_C	48.8	SRL
Marginalized, IG_B	58.9	SRL

Table 8: Comparison of grammar induction approaches. We contrast the DMV trained with Viterbi EM+uniform initialization (DMV), our constrained DMV (DMV+C), and our model's MBR decoding of latent syntax (Marginalized) with other recent work: Spitkovsky et al. (2010a) (SAJM'10), Spitkovsky et al. (2010b) (SJA'10), Naseem and Barzilay (2011) (NB'11), and the CS model of Spitkovsky et al. (2013) (SAJ'13).

This suggests that refining the low-resource grammar induction methods may lead to gains in SRL.

Interestingly, the marginalized grammars best the DMV grammar induction method; however, this difference is less pronounced when the DMV is constrained using SRL labels as distant supervision. This could indicate that a better model for grammar induction would result in better performance for SRL. We therefore turn to an analysis of other approaches to grammar induction in Table 8, evaluated on the Penn Treebank. We contrast with methods using distant supervision (Naseem and Barzilay, 2011; Spitkovsky et al., 2010b) and fully unsupervised dependency parsing (Spitkovsky et al., 2013). Following prior work, we exclude punctuation from evaluation and convert the constituency trees to dependencies.[12]

The approach from Spitkovsky et al. (2013) (SAJ'13) outperforms all other approaches, including our marginalized settings. We therefore may be able to achieve further gains in the pipeline model by considering better models of latent syntax, or better search techniques that break out of local optima. Similarly, improving the non-convex optimization of our latent-variable CRF (Marginalized) may offer further gains.

5 Discussion and Future Work

We have compared various approaches for low-resource semantic role labeling at the state-of-the-art level. We find that we can outperform prior work in the low-resource setting by coupling the selection of feature templates based on information gain with a joint model that marginalizes over latent syntax.

We utilize unlabeled data in both generative and discriminative models for dependency syntax and in generative word clustering. Our discriminative joint models treat latent syntax as a structured-feature to be optimized for the end-task of SRL, while our other grammar induction techniques optimize for unlabeled data likelihood—optionally with distant supervision. We observe that careful use of these unlabeled data resources can improve performance on the end task.

Our subtractive experiments suggest that lemma annotations, a high-resource annotation, may not provide a large benefit for SRL. Our grammar induction analysis indicates that relatively low accuracy can still result in reasonable SRL predictions; still, the models do not outperform those that use supervised syntax, and we aim to explore how well the pipeline models in particular improve when we apply higher accuracy unsupervised grammar induction techniques.

We have utilized well studied datasets in order to best understand the quality of our models relative to prior work. In future work, we hope to explore the effectiveness of our approaches on truly low resource settings by using crowdsourcing to develop semantic role datasets in other languages and domains.

Acknowledgments We thank Richard Johansson, Dennis Mehay, and Stephen Boxwell for help with data. We also thank Jason Naradowsky, Jason Eisner, and anonymous reviewers for comments on the paper.

[12]Naseem and Barzilay (2011) and our results use the Penn converter (Pierre and Heiki-Jaan, 2007). Spitkovsky et al. (2010b; 2013) use Collins (1999) head percolation rules.

References

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling*. Prentice-Hall, Inc.

Rami Al-Rfou', Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of the 17th Conference on Computational Natural Language Learning (CoNLL 2013)*. Association for Computational Linguistics.

Anders Björkelund, Love Hafdell, and Pierre Nugues. 2009. Multilingual semantic role labeling. In *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*. Association for Computational Linguistics.

Stephen Boxwell and Michael White. 2008. Projecting propbank roles onto the CCGbank. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2008)*. European Language Resources Association.

Stephen Boxwell, Chris Brew, Jason Baldridge, Dennis Mehay, and Sujith Ravi. 2011. Semantic role labeling without treebanks? In *Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP)*. Asian Federation of Natural Language Processing.

Peter F. Brown, Peter V. Desouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4).

Trevor Cohn, Phil Blunsom, and Sharon Goldwater. 2010. Inducing tree-substitution grammars. *The Journal of Machine Learning Research*, 11.

Michael Collins. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, University of Pennsylvania.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*. Association for Computational Linguistics.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3).

Richard Johansson and Pierre Nugues. 2008. Dependency-based semantic role labeling of PropBank. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP 2008)*. Association for Computational Linguistics.

Dan Klein and Christopher Manning. 2004. Corpus-Based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL 2004)*. Association for Computational Linguistics.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*. Association for Computational Linguistics.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML 2001)*. Morgan Kaufmann.

Xavier Lluís, Xavier Carreras, and Lluís Màrquez. 2013. Joint arc-factored parsing of syntactic and semantic dependencies. *Transactions of the Association for Computational Linguistics (TACL)*.

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

Andre Martins, Noah Smith, Mario Figueiredo, and Pedro Aguiar. 2011. Structured sparsity in structured prediction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*. Association for Computational Linguistics.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*. Association for Computational Linguistics.

Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In Susan Dumais, Daniel Marcu, and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*. Association for Computational Linguistics.

Jason Naradowsky, Sebastian Riedel, and David Smith. 2012. Improving NLP through marginalization of hidden syntactic structure. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing (EMNLP 2012)*. Association for Computational Linguistics.

Tahira Naseem and Regina Barzilay. 2011. Using semantic cues to learn syntax. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI 2011)*. AAAI Press.

Fernando Pereira and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics (ACL 1992)*.

Nugues Pierre and Kalep Heiki-Jaan. 2007. Extended constituent-to-dependency conversion for english. *NODALIDA 2007 Proceedings*.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2).

David A. Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2008)*. Association for Computational Linguistics.

Valentin I. Spitkovsky, Hiyan Alshawi, Daniel Jurafsky, and Christopher D Manning. 2010a. Viterbi training improves unsupervised dependency parsing. In *Proceedings of the 14th Conference on Computational Natural Language Learning (CoNLL 2010)*. Association for Computational Linguistics.

Valentin I. Spitkovsky, Daniel Jurafsky, and Hiyan Alshawi. 2010b. Profiting from mark-up: Hyper-text annotations for guided parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)*. Association for Computational Linguistics.

Valentin I. Spitkovsky, Hiyan Alshawi, Angel X. Chang, and Daniel Jurafsky. 2011. Unsupervised dependency parsing without gold part-of-speech tags. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*. Association for Computational Linguistics.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2013. Breaking out of local optima with count transforms and model recombination: A study in grammar induction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*. Association for Computational Linguistics.

Kristina Toutanova, Aria Haghighi, and Christopher Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL 2005)*. Association for Computational Linguistics.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature hashing for large scale multitask learning. In Léon Bottou and Michael Littman, editors, *Proceedings of the 26th Annual International Conference on Machine Learning (ICML 2009)*. Omnipress.

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

Hai Zhao, Wenliang Chen, Chunyu Kity, and Guodong Zhou. 2009. Multilingual dependency learning: A huge feature engineering method to semantic dependency parsing. In *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*. Association for Computational Linguistics.

Joint Syntactic and Semantic Parsing with Combinatory Categorial Grammar

Jayant Krishnamurthy
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
jayantk@cs.cmu.edu

Tom M. Mitchell
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
tom.mitchell@cmu.edu

Abstract

We present an approach to training a joint syntactic and semantic parser that combines syntactic training information from CCGbank with semantic training information from a knowledge base via distant supervision. The trained parser produces a full syntactic parse of any sentence, while simultaneously producing logical forms for portions of the sentence that have a semantic representation within the parser's predicate vocabulary. We demonstrate our approach by training a parser whose semantic representation contains 130 predicates from the NELL ontology. A semantic evaluation demonstrates that this parser produces logical forms better than both comparable prior work and a pipelined syntax-then-semantics approach. A syntactic evaluation on CCGbank demonstrates that the parser's dependency F-score is within 2.5% of state-of-the-art.

1 Introduction

Integrating syntactic parsing with semantics has long been a goal of natural language processing and is expected to improve both syntactic and semantic processing. For example, semantics could help predict the differing prepositional phrase attachments in "I caught the butterfly with the net" and "I caught the butterfly with the spots." A joint analysis could also avoid propagating syntactic parsing errors into semantic processing, thereby improving performance.

We suggest that a large populated knowledge base should play a key role in syntactic and semantic parsing: in training the parser, in resolving syntactic ambiguities when the trained parser is applied to new text, and in its output semantic representation. Using semantic information from the knowledge base at training and test time will ideally improve the parser's ability to solve difficult syntactic parsing problems, as in the examples above. A semantic representation tied to a knowledge base allows for powerful inference operations – such as identifying the possible entity referents of a noun phrase – that cannot be performed with shallower representations (e.g., frame semantics (Baker et al., 1998) or a direct conversion of syntax to logic (Bos, 2005)).

This paper presents an approach to training a joint syntactic and semantic parser using a large background knowledge base. Our parser produces a full syntactic parse of every sentence, and furthermore produces logical forms for portions of the sentence that have a semantic representation within the parser's predicate vocabulary. For example, given a phrase like "my favorite town in California," our parser will assign a logical form like $\lambda x.\text{CITY}(x) \wedge \text{LOCATEDIN}(x, \text{CALIFORNIA})$ to the "town in California" portion. Additionally, the parser uses predicate and entity type information during parsing to select a syntactic parse.

Our parser is trained by combining a syntactic parsing task with a distantly-supervised relation extraction task. Syntactic information is provided by CCGbank, a conversion of the Penn Treebank into the CCG formalism (Hockenmaier and Steedman, 2002a). Semantics are learned by training the parser to extract knowledge base relation instances from a corpus of unlabeled sentences, in a distantly-supervised training regime. This approach uses the knowledge base to avoid expensive manual labeling of individual sentence semantics. By optimizing the parser to perform both tasks simultaneously, we train a parser that produces accurate syntactic and semantic analyses.

We demonstrate our approach by training a joint syntactic and semantic parser, which we call ASP. ASP produces a full syntactic analysis of every sentence while simultaneously producing logical forms containing any of 61 category and 69 re-

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1188–1198,
Baltimore, Maryland, USA, June 23-25 2014.

lation predicates from NELL. Experiments with ASP demonstrate that jointly analyzing syntax and semantics improves semantic parsing performance over comparable prior work and a pipelined syntax-then-semantics approach. ASP's syntactic parsing performance is within 2.5% of state-of-the-art; however, we also find that incorporating semantic information reduces syntactic parsing accuracy by $\sim 0.5\%$.

2 Prior Work

This paper combines two lines of prior work: broad coverage syntactic parsing with CCG and semantic parsing.

Broad coverage syntactic parsing with CCG has produced both resources and successful parsers. These parsers are trained and evaluated using CCGbank (Hockenmaier and Steedman, 2002a), an automatic conversion of the Penn Treebank into the CCG formalism. Several broad coverage parsers have been trained using this resource (Hockenmaier and Steedman, 2002b; Hockenmaier, 2003b). The parsing model in this paper is loosely based on C&C (Clark and Curran, 2007b; Clark and Curran, 2007a), a discriminative log-linear model for statistical parsing. Some work has also attempted to automatically derive logical meaning representations directly from syntactic CCG parses (Bos, 2005; Lewis and Steedman, 2013). However, these approaches to semantics do not ground the text to beliefs in a knowledge base.

Meanwhile, work on semantic parsing has focused on producing semantic parsers for answering simple natural language questions (Zelle and Mooney, 1996; Ge and Mooney, 2005; Wong and Mooney, 2006; Wong and Mooney, 2007; Lu et al., 2008; Kate and Mooney, 2006; Zettlemoyer and Collins, 2005; Kwiatkowski et al., 2011). This line of work has typically used a corpus of sentences with annotated logical forms to train the parser. Recent work has relaxed the requisite supervision conditions (Clarke et al., 2010; Liang et al., 2011), but has still focused on simple questions. Finally, some work has looked at applying semantic parsing to answer queries against large knowledge bases, such as YAGO (Yahya et al., 2012) and Freebase (Cai and Yates, 2013b; Cai and Yates, 2013a; Kwiatkowski et al., 2013; Berant et al., 2013). Although this work considers a larger number (thousands) of predicates than we do, none of these systems are capable of parsing open-domain text. Our approach is most closely related to the distantly-supervised approach of Krishnamurthy and Mitchell (2012).

The parser presented in this paper can be viewed as a combination of both a broad coverage syntactic parser and a semantic parser trained using distant supervision. Combining these two lines of work has synergistic effects – for example, our parser is capable of semantically analyzing conjunctions and relative clauses based on the syntactic annotation of these categories in CCGbank. This synergy gives our parser a richer semantic representation than previous work, while simultaneously enabling broad coverage.

3 Parser Design

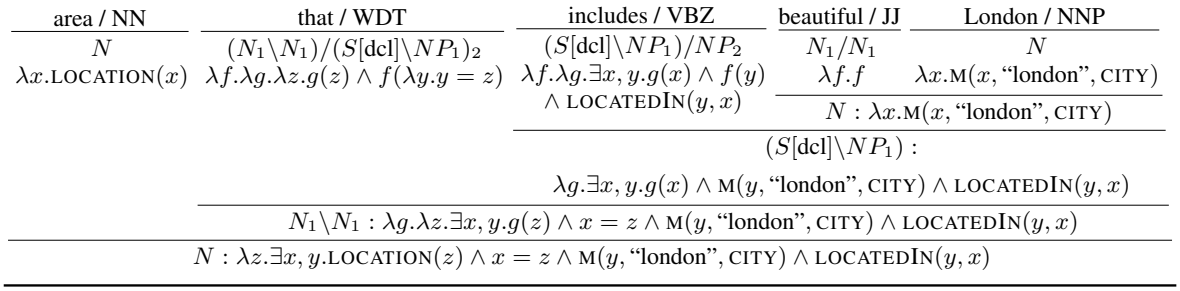
This section describes the Combinatory Categorial Grammar (CCG) parsing model used by ASP. The input to the parser is a part-of-speech tagged sentence, and the output is a syntactic CCG parse tree, along with zero or more logical forms representing the semantics of subspans of the sentence. These logical forms are constructed using category and relation predicates from a broad coverage knowledge base. The parser also outputs a collection of dependency structures summarizing the sentence's predicate-argument structure. Figure 1 illustrates ASP's input/output specification.

3.1 Knowledge Base

The parser uses category and relation predicates from a broad coverage knowledge base both to construct logical forms and to parametrize the parsing model. The knowledge base is assumed to have two kinds of ontological structure: a generalization/subsumption hierarchy and argument type constraints. This paper uses NELL's ontology (Carlson et al., 2010), which, for example, specifies that the category ORGANIZATION is a generalization of SPORTSTEAM, and that both arguments to the LOCATEDIN relation must have type LOCATION. These type constraints are enforced during parsing. Throughout this paper, predicate names are shown in SMALLCAPS.

3.2 Syntax

ASP uses a lexicalized and semantically-typed Combinatory Categorial Grammar (CCG) (Steedman, 1996). Most grammatical information in CCG is encoded in a lexicon Λ, containing entries such as:



		Head				Argument			
word	POS	semantic type	index	syntactic category	arg. num.	word	POS	semantic type	index
that	WDT	—	1	$(N_1\backslash N_1)/(S\backslash NP_1)_2$	1	area	NN	LOCATION	0
that	WDT	—	1	$(N_1\backslash N_1)/(S\backslash NP_1)_2$	2	includes	VBZ	LOCATEDIN^{-1}	2
includes	VBZ	LOCATEDIN^{-1}	2	$(S[\text{dcl}]\backslash NP_1)/NP_2$	1	area	NN	LOCATION	0
includes	VBZ	LOCATEDIN^{-1}	2	$(S[\text{dcl}]\backslash NP_1)/NP_2$	2	ENTITY:CITY	NNP	CITY	4
beautiful	JJ	—	3	N_1/N_1	1	ENTITY:CITY	NNP	CITY	4

Figure 1: Example input and output for ASP. Given a POS-tagged sentence, the parser produces a CCG syntactic tree and logical form (top), and a collection of dependency structures (bottom).

person := N : PERSON : $\lambda x.\text{PERSON}(x)$

London := N : CITY : $\lambda x.\text{M}(x, \text{“london”}, \text{CITY})$

great := N_1/N_1 : — : $\lambda f.\lambda x.f(x)$

bought := $(S[\text{dcl}]\backslash NP_1)/NP_2$: ACQUIRED : $\lambda f.\lambda g.\exists x, y.f(y) \wedge g(x) \wedge \text{ACQUIRED}(x, y)$

Each lexicon entry maps a word to a syntactic category, semantic type, and logical form. CCG has two kinds of syntactic categories: atomic and functional. Atomic categories include N for noun and S for sentence. Functional categories are functions constructed recursively from atomic categories; these categories are denoted using slashes to separate the category's argument type from its return type. The argument type appears on the right side of the slash, and the return type on the left. The direction of slash determines where the argument must appear – / means an argument on the right, and \ means an argument on the left.

Syntactic categories in ASP are annotated with two additional kinds of information. First, atomic categories may have associated syntactic features given in square brackets. These features are used in CCGbank to distinguish variants of atomic syntactic categories, e.g., $S[\text{dcl}]$ denotes a declarative sentence. Second, each category is annotated with head and dependency information using subscripts. These subscripts are used to populate predicate-argument dependencies (described below), and to pass head information using unification. For example, the head of the parse in Figure 1 is "area," due to the coindexing of the argument and return categories in the category $N_1\backslash N_1$.

In addition to the syntactic category, each lexicon entry has a semantic type and a logical form. The semantic type is a category or relation predicate that concisely represents the word's semantics. The semantic type is used to enforce type constraints during parsing and to include semantics in the parser's parametrization. The logical form gives the full semantics of the word in lambda calculus. The parser also allows lexicon entries with the semantic type "—", representing words whose semantics cannot be expressed using predicates from the ontology.

Parsing in CCG combines adjacent categories using a small number of combinators, such as function application:

$$X/Y : f \quad Y : g \implies X : f(g)$$
$$Y : g \quad X\backslash Y : f \implies X : f(g)$$

The first rule states that the category X/Y can be applied to the category Y, returning category X, and that the logical form f is applied to g to produce the logical form for the returned category. Head words and semantic types are also propagated to the returned category based on the annotated head-passing markup.

3.3 Dependency Structures

Parsing a sentence produces a collection of dependency structures which summarize the predicate-argument structure of the sentence. Dependency structures are 10-tuples, of the form:

< head word, head POS, head semantic type, head word index, head word syntactic category, argument number, argument word, argument POS, argument semantic type, argument word index >

A dependency structure captures a relationship between a head word and its argument. During parsing, whenever a subscripted argument of a syntactic category is filled, a dependency structure

is created between the head of the applied function and its argument. For example, in Figure 1, the first application fills argument 1 of "beautiful" with "London," creating a dependency structure.

3.4 Logical Forms

ASP performs a best-effort semantic analysis of every parsed sentence, producing logical forms for subspans of the sentence when possible. Logical forms are designed so that the meaning of a sentence is a universally- and existentially-quantified conjunction of predicates with partially shared arguments. This representation allows the parser to produce semantic analyses for a reasonable subset of language, including prepositions, verbs, nouns, relative clauses, and conjunctions.

Figure 1 shows a representative sample of a logical form produced by ASP. Generally, the parser produces a lambda calculus statement with several existentially-quantified variables ranging over entities in the knowledge base. The only exception to this rule is conjunctions, which are represented using a scoped universal quantifier over the conjoined predicates. Entity mentions appear in logical forms via a special mention predicate, M, instead of as database constants. For example, "London" appears as $\text{M}(x, \text{"london"}, \text{CITY})$, instead of as a constant like LONDON. The meaning of this mention predicate is that x is an entity which can be called "london" and belongs to the CITY category. This representation propagates uncertainty about entity references into the logical form where background knowledge can be used for disambiguation. For example, "London, England" is assigned a logical form that disambiguates "London" to a "London" located in "England."[1]

Lexicon entries without a semantic type are automatically assigned logical forms based on their head passing markup. For example, in Figure 1, the adjective "beautiful" is assigned $\lambda f.f$. This approach allows a logical form to be derived for most sentences, but (somewhat counterintuitively) can lose interesting logical forms from constituent subspans. For example, the preposition "in" has syntactic category $(N_1 \backslash N_1)/N_2$, which results in the logical form $\lambda f.\lambda g.g$. This logical form discards any information present in the argument f. We avoid this problem by extracting a logical form from every subtree of the CCG parse.

[1] Specifically, $\lambda x.\exists y.\text{CITYLOCATEDINCOUNTRY}(x, y) \wedge \text{M}(x, \text{"london"}, \text{CITY}) \wedge \text{M}(y, \text{"england"}, \text{COUNTRY})$

3.5 Parametrization

The parser Γ is trained as a discriminative linear model of the following form:

$$\Gamma(\ell, d, t|s; \theta) \quad = \quad \theta^T \phi(d, t, s)$$

Given a parameter vector θ and a sentence s, the parser produces a score for a syntactic parse tree t, a collection of dependency structures d and a logical form ℓ. The score depends on features of the parse produced by the feature function ϕ.

ϕ contains four classes of features: lexicon features, combinator features, dependency features and dependency distance features (Table 1). These features are based on those of C&C (Clark and Curran, 2007b), modified to include semantic types. The features are designed to share syntactic information about a word across its distinct semantic realizations in order to transfer syntactic information from CCGbank to semantic parsing.

The parser also includes a hard type-checking constraint to ensure that logical forms are well-typed. This constraint states that dependency structures with a head semantic type only accept arguments that (1) have a semantic type, and (2) are within the domain/range of the head type.

4 Parameter Estimation

This section describes the training procedure for ASP. Training is performed by minimizing a joint objective function combining a syntactic parsing task and a distantly-supervised relation extraction task. The input training data includes:

1. A collection L of sentences s_i with annotated syntactic trees t_i (e.g., CCGbank).
2. A corpus of sentences S (e.g., Wikipedia).
3. A knowledge base K (e.g., NELL), containing relation instances $r(e_1, e_2) \in K$.
4. A CCG lexicon Λ (see Section 5.2).

Given these resources, the algorithm described in this section produces parameters θ for a semantic parser. Our parameter estimation procedure constructs a joint objective function $O(\theta)$ that decomposes into syntactic and semantic components: $O(\theta) = O_{\text{syn}}(\theta) + O_{\text{sem}}(\theta)$. The syntactic component O_{syn} is a standard syntactic parsing objective constructed using the syntactic resource L. The semantic component O_{sem} is a distantly-supervised relation extraction task based on the semantic constraint from Krishnamurthy and Mitchell (2012). These components are described in more detail in the following sections.

Lexicon features: word, POS $:= X : t : \ell$

Word/syntactic category	word, X
POS/syntactic category	POS, X
Word semantics	word, X, t

Combinator features: $X\ Y \rightarrow Z$ or $X \rightarrow Z$

Binary combinator indicator	$X\ Y \rightarrow Z$
Unary combinator indicator	$X \rightarrow Z$
Root syntactic category	Z

Dependency Features: $< h_w, h_p, h_t, h_i, s, n, a_w, a_p, a_t, a_i >$

Predicate-Argument Indicator	$< h_w, —, h_t, —, s, n, a_w, —, a_t, — >$
Word-Word Indicator	$< h_w, —, —, —, s, n, a_w, —, —, — >$
Predicate-POS Indicator	$< h_w, —, h_t, —, s, n, —, a_p, —, — >$
Word-POS Indicator	$< h_w, —, —, —, s, n, —, a_p, —, — >$
POS-Argument Indicator	$< —, h_p, —, —, s, n, a_w, —, a_t, — >$
POS-Word Indicator	$< —, h_p, —, —, s, n, a_w, —, —, — >$
POS-POS Indicator	$< —, h_p, —, —, s, n, —, a_p, —, — >$

Dependency Distance Features:

Token distance	$h_w, h_t, —, s, n, d$	d = Number of tokens between h_i and a_i: 0, 1, 2 or more.
Token distance word backoff	$h_w, —, s, n, d$	d = Number of tokens between h_i and a_i: 0, 1, 2 or more.
Token distance POS backoff	$—, —, h_p, s, n, d$	d = Number of tokens between h_i and a_i: 0, 1, 2 or more.

(The above distance features are repeated using the number of intervening verbs and punctuation marks.)

Table 1: Listing of parser feature templates used in the feature function ϕ. Each feature template represents a class of indicator features that fire during parsing when lexicon entries are used, combinators are applied, or dependency structures are instantiated.

4.1 Syntactic Objective

The syntactic objective is the structured perceptron objective instantiated for a syntactic parsing task. This objective encourages the parser to accurately reproduce the syntactic parses in the annotated corpus $L = \{(s_i, t_i)\}_{i=1}^n$:

$$O_{\text{syn}}(\theta) = \sum_{i=1}^{n} | \max_{\hat{\ell},\hat{d},\hat{t}} \Gamma(\hat{\ell}, \hat{d}, \hat{t} | s_i; \theta) - \max_{\ell^*, d^*} \Gamma(\ell^*, d^*, t_i | s_i; \theta) |_+$$

The first term in the above expression represents the best CCG parse of the sentence s_i according to the current model. The second term is the best parse of s_i whose syntactic tree equals the true syntactic tree t_i. In the above equation $|\cdot|_+$ denotes the positive part of the expression. Minimizing this objective therefore finds parameters θ that reproduce the annotated syntactic trees.

4.2 Semantic Objective

The semantic objective corresponds to a distantly-supervised relation extraction task that constrains the logical forms produced by the semantic parser. Distant supervision is provided by the following constraint: every relation instance $r(e_1, e_2) \in K$ must be expressed by at least one sentence in $\mathbf{S}_{(e_1,e_2)}$, the set of sentences that mention both e_1 and e_2 (Hoffmann et al., 2011). If this constraint is empirically true and sufficiently constrains the parser's logical forms, then optimizing the semantic objective produces an accurate semantic parser.

A training example in the semantic objective consists of the set of sentences mentioning a pair of entities, $\mathbf{S}_{(e_1,e_2)} = \{s_1, s_2, ...\}$, paired with a binary vector representing the set of relations that the two entities participate in, $\mathbf{y}_{(e_1,e_2)}$. The distant supervision constraint Ψ forces the logical forms predicted for the sentences to entail the relations in $\mathbf{y}_{(e_1,e_2)}$. Ψ is a deterministic OR constraint that checks whether each logical form entails the relation instance $r(e_1, e_2)$, deterministically setting $y_r = 1$ if any logical form entails the instance and $y_r = 0$ otherwise.

Let $(\boldsymbol{\ell}, \boldsymbol{d}, \mathbf{t})$ represent a collection of semantic parses for the sentences $\mathbf{S} = \mathbf{S}_{(e_1,e_2)}$. Let $\Gamma(\boldsymbol{\ell}, \boldsymbol{d}, \mathbf{t} | \mathbf{S}; \theta) = \sum_{i=1}^{|\mathbf{S}|} \Gamma(\ell_i, d_i, t_i | s_i; \theta)$ represent the total weight assigned by the parser to a collection of parses for the sentences $\mathbf{S}$. For the pair of entities (e_1, e_2), the semantic objective is:

$$O_{\text{sem}}(\theta) = | \max_{\hat{\boldsymbol{\ell}},\hat{\boldsymbol{d}},\hat{\mathbf{t}}} \Gamma(\hat{\boldsymbol{\ell}}, \hat{\boldsymbol{d}}, \hat{\mathbf{t}} | \mathbf{S}; \theta) - \max_{\boldsymbol{\ell}^*, \boldsymbol{d}^*, \mathbf{t}^*} \big(\Psi(\mathbf{y}_{(e_1,e_2)}, \boldsymbol{\ell}^*, \boldsymbol{d}^*, \mathbf{t}^*) + \Gamma(\boldsymbol{\ell}^*, \boldsymbol{d}^*, \mathbf{t}^* | \mathbf{S}; \theta)\big) |_+$$

4.3 Optimization

Training minimizes the joint objective using the structured perceptron algorithm, which can be viewed as the stochastic subgradient method (Ratliff et al., 2006) applied to the objective $O(\theta)$. We initialize the parameters to zero, i.e., $\theta^0 = 0$. On each iteration, we sample either a syntactic example (s_i, t_i) or a semantic example $(\mathbf{S}_{(e_1,e_2)}, \mathbf{y}_{(e_1,e_2)})$. If a syntactic example is sampled, we apply the following parameter update:

$$\begin{aligned} \hat{\ell}, \hat{d}, \hat{t} &\leftarrow \arg\max_{\ell,d,t} \Gamma(\ell, d, t | s_i; \theta^t) \\ \ell^*, d^* &\leftarrow \arg\max_{\ell,d} \Gamma(\ell, d, t_i | s_i; \theta^t) \\ \theta^{t+1} &\leftarrow \theta^t + \phi(d^*, t_i, s_i) - \phi(\hat{d}, \hat{t}, s_i) \end{aligned}$$

This update moves the parameters toward the features of the best parse with the correct syntactic derivation, $\phi(d^*, t_i, s_i)$. If a semantic example is

	Labeled Dependencies			Unlabeled Dependencies			
	P	R	F	P	R	F	Coverage
ASP	85.58	85.31	85.44	91.75	91.46	91.60	99.63
ASP-SYN	86.06	85.84	85.95	92.13	91.89	92.01	99.63
C&C (Clark and Curran, 2007b)	88.34	86.96	87.64	93.74	92.28	93.00	99.63
(Hockenmaier, 2003a)	84.3	84.6	84.4	91.8	92.2	92.0	99.83

Table 2: Syntactic parsing results for Section 23 of CCGbank. Parser performance is measured using precision (P), recall (R) and F-measure (F) of labeled and unlabeled dependencies.

sampled, we instead apply the following update:

$$\hat{\boldsymbol{\ell}}, \hat{\boldsymbol{d}}, \hat{\mathbf{t}} \leftarrow \arg\max_{\boldsymbol{\ell}, \boldsymbol{d}, \mathbf{t}} \Gamma(\boldsymbol{\ell}, \boldsymbol{d}, \mathbf{t} | \mathbf{S}_{(e_1,e_2)}; \theta^t)$$

$$\boldsymbol{\ell}^*, \boldsymbol{d}^*, \mathbf{t}^* \leftarrow \arg\max_{\boldsymbol{\ell}, \boldsymbol{d}, \mathbf{t}} \Gamma(\boldsymbol{\ell}, \boldsymbol{d}, \mathbf{t} | \mathbf{S}_{(e_1,e_2)}; \theta^t) + \Psi(\mathbf{y}_{(e_1,e_2)}, \boldsymbol{\ell}, \boldsymbol{d}, \mathbf{t})$$

$$\theta^{t+1} \leftarrow \theta^t + \phi(\boldsymbol{d}^*, \mathbf{t}^*, \mathbf{S}_{(e_1,e_2)}) - \phi(\hat{\boldsymbol{d}}, \hat{\mathbf{t}}, \mathbf{S}_{(e_1,e_2)})$$

This update moves the parameters toward the features of the best set of parses that satisfy the distant supervision constraint. Training outputs the average of each iteration's parameters, $\bar{\theta} = \frac{1}{n} \sum_{t=1}^{n} \theta^t$. In practice, we train the parser by performing a single pass over the examples in the data set.

All of the maximizations above can be performed exactly using a CKY-style chart parsing algorithm, except for the last one. This maximization is intractable due to the coupling between logical forms in $\boldsymbol{\ell}$ caused by enforcing the distant supervision constraint. We approximate this maximization in two steps. First, we perform a beam search to produce a list of candidate parses for each sentence $s \in \mathbf{S}_{(e_1,e_2)}$. We then extract relation instances from each parse and apply the greedy inference algorithm from Hoffmann et al., (2011) to identify the best set of parses that satisfy the distant supervision constraint. The procedure skips any examples with sentences that cannot be parsed (due to beam search failures) or where the distant supervision constraint cannot be satisfied.

5 Experiments

The experiments below evaluate ASP's syntactic and semantic parsing ability. The parser is trained on CCGbank and a corpus of Wikipedia sentences, using NELL's predicate vocabulary. The syntactic analyses of the trained parser are evaluated against CCGbank, and its logical forms are evaluated on an information extraction task and against an annotated test set of Wikipedia sentences.

5.1 Data Sets

The data sets for the evaluation consist of CCGbank, a corpus of dependency-parsed Wikipedia sentences, and a logical knowledge base derived from NELL and Freebase. Sections 02-21 of CCGbank were used for training, Section 00 for validation, and Section 23 for the final results. The knowledge base's predicate vocabulary is taken from NELL, and its instances are taken from Freebase using a manually-constructed mapping between Freebase and NELL. Using Freebase relation instances produces cleaner training data than NELL's automatically-extracted instances.

Using the relation instances and Wikipedia sentences, we constructed a data set for distantly-supervised relation extraction. We identified mentions of entities in each sentence using simple string matching, then aggregated these sentences by entity pair. 20% of the entity pairs were set aside for validation. In the remaining training data, we downsampled entity pairs that did not participate in at least one relation. We further eliminated sentences containing more than 30 tokens. The resulting training corpus contains 25k entity pairs (half of which participate in a relation), 41k sentences, and 71 distinct relation predicates.

5.2 Grammar Construction

The grammar for ASP contains the annotated lexicon entries and grammar rules in Sections 02-21 of CCGbank, and additional semantic entries produced using a set of dependency parse heuristics.

The lexicon Λ contains all words that occur at least 20 times in CCGbank. Rare words are replaced by their part of speech. The head passing and dependency markup was generated using the rules of the C&C parser (Clark and Curran, 2007b). These lexicon entries are also annotated with logical forms capturing their head passing relationship. For example, the adjective category N_1/N_1 is annotated with the logical form $\lambda f.f.$ These entries are all assigned semantic type —.

We augment this lexicon with additional entries

Sentence	Extracted Logical Form
St. John, a Mexican-American born in San Francisco, California, her family comes from Zacatecas, Mexico.	$\lambda x.\exists y, z.\text{M}(x, \text{“st. john”}) \wedge \text{M}(y, \text{“san francisco”}) \wedge \text{PERSONBORNINLOCATION}(x, y) \wedge \text{CITYLOCATEDINSTATE}(y, z) \wedge \text{M}(z, \text{“california”})$
The capital and largest city of Laos is Vientiane and other major cities include Luang Prabang, Savannakhet and Pakse.	$\exists x, y.\text{M}(x, \text{“vientiane”}) \wedge \text{CITY}(x) \wedge \text{CITYCAPITALOFCOUNTRY}(x, y) \wedge \text{M}(y, \text{“laos”})$
Gellar next played a lead role in James Toback 's critically unsuccessful independent “Harvard Man” (2001), where she played the daughter of a mobster.	$\lambda x.\exists y.\text{M}(y, \text{“james toback”}) \wedge \text{DIRECTORDIRECTEDMOVIE}(y, x) \wedge \text{M}(x, \text{“harvard man”})$

Figure 2: Logical forms produced by ASP for sentences in the information extraction corpus. Each logical form is extracted from the underlined sentence portion.

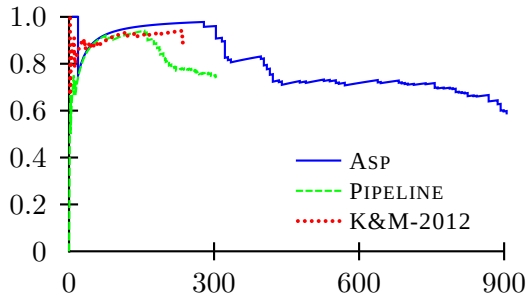


Figure 3: Logical form precision as a function of the expected number of correct extracted logical forms. ASP extracts more correct logical forms because it jointly analyzes syntax and semantics.

mapping words to logical forms with NELL predicates. These entries are instantiated using a set of dependency parse patterns, listed in an online appendix.[2] These patterns are applied to the training corpus, heuristically identifying verbs, prepositions, and possessives that express relations, and nouns that express categories. The patterns also include special cases for forms of “to be.” This process generates ∼4000 entries (not counting entity names), representing 69 relations and 61 categories from NELL. Section 3.2 shows several lexicon entries generated by this process.

The parser's combinators include function application, composition, and crossed composition, as well as several binary and unary type-changing rules that occur in CCGbank. All combinators were restricted to only apply to categories that combine in Sections 02-21. Finally, the grammar includes a number of heuristically-instantiated binary rules of the form , $N \rightarrow N\backslash N$ that instantiate a relation between adjacent nouns. These rules capture appositives and some other constructions.

5.3 Supertagging

Parsing in practice can be slow because the parser's lexicalized grammar permits a large number of parses for a sentence. We improve parser performance by performing *supertagging* (Bangalore and Joshi, 1999; Clark and Curran, 2004). We trained a logistic regression classifier to predict the syntactic category of each token in a sentence from features of the surrounding tokens and POS tags. Subsequent parsing is restricted to only consider categories whose probability is within a factor of α of the highest-scoring category. The parser uses a backoff strategy, first attempting to parse with the supertags from $\alpha = 0.01$, backing off to $\alpha = 0.001$ if the initial parsing attempt fails.

5.4 Syntactic Evaluation

The syntactic evaluation measures ASP's ability to reproduce the predicate-argument dependencies in CCGbank. As in previous work, our evaluation uses *labeled* and *unlabeled* dependencies. Labeled dependencies are dependency structures with both words and semantic types removed, leaving two word indexes, a syntactic category, and an argument number. Unlabeled dependencies further eliminate the syntactic category and argument number, leaving a pair of word indexes. Performance is measured using precision, recall, and F-measure against the annotated dependency structures in CCGbank. Precision is the fraction of predicted dependencies which are in CCGbank, recall is the fraction of CCGbank dependencies produced by the parser, and F-measure is the harmonic mean of precision and recall.

For comparison, we also trained a syntactic version of our parser, ASP-SYN, using only the CCGbank lexicon and grammar. Comparing against this parser lets us measure the effect of the relation extraction task on syntactic parsing.

Table 2 shows the results of our evaluation. For comparison, we include results for two existing syntactic CCG parsers: C&C, the current state-of-the-art CCG parser (Clark and Curran, 2007b), and the next best system (Hockenmaier, 2003a). Both ASP and ASP-SYN perform reasonably well, within 2.5% of the performance of C&C at the same coverage level. However, ASP-

[2] http://rtw.ml.cmu.edu/acl2014_asp/

	Logical Form Accuracy	Extraction Precision	Extraction Recall
ASP	0.28	0.90	0.32
K&M-2012	0.14	1.00	0.06
PIPELINE	0.2	0.63	0.17

Table 3: Logical form accuracy and extraction precision/recall on the annotated test set. The high extraction recall for ASP shows that it produces more complete logical forms than either baseline.

SYN outperforms ASP by around 0.5%, suggesting that ASP's additional semantic knowledge slightly hurts syntactic parsing performance. This performance loss appears to be largely due to poor entity mention detection, as we found that not using entity mention lexicon entries at test time improves ASP's labeled and unlabeled F-scores by 0.3% on Section 00. The knowledge base contains many infrequently-mentioned entities with common names; these entities contribute incorrect semantic type information that confuses the parser.

5.5 Semantic Evaluation

We performed two semantic evaluations to better understand ASP's ability to construct logical forms. The first evaluation emphasizes precision over recall, and the second evaluation accurately measures recall using a manually labeled test set.

5.5.1 Baselines

For comparison, we also trained two baseline models. The first baseline, PIPELINE, is a pipelined syntax-then-semantics approach designed to mimic Boxer (Bos, 2005). This baseline first syntactically parses each sentence using ASP-SYN, then produces a semantic analysis by assigning a logical form to each word. We train this baseline using the semantic objective (Section 4.2) while holding fixed the syntactic parse of each sentence. Note that, unlike Boxer, this baseline learns which logical form to assign each word, and its logical forms contain NELL predicates.

The second baseline, K&M-2012, is the approach of Krishnamurthy and Mitchell (2012), representing the state-of-the-art in distantly-supervised semantic parsing. This approach trains a semantic parser by combining distant semantic supervision with syntactic supervision from dependency parses. The best performing variant of this system also uses dependency parses at test time to constrain the interpretation of test sentences – hence, this system also uses a pipelined syntax-then-semantics approach. To improve comparability, we reimplemented this approach using our parsing model, which has richer features than were used in their paper.

5.5.2 Information Extraction Evaluation

The information extraction evaluation uses each system to extract logical forms from a large corpus of sentences, then measures the fraction of extracted logical forms that are correct. The test set consists of 8.5k sentences sampled from the held-out Wikipedia sentences. Each system was run on this data set, extracting all logical forms from each sentence that entailed at least one category or relation instance. We ranked these extractions using the parser's inside chart score, then manually annotated a sample of 250 logical forms from each system for correctness. Logical forms were marked correct if all category and relation instances entailed by the logical form were expressed by the sentence. Note that a correct logical form need not entail all of the relations expressed by the sentence, reflecting an emphasis on precision over recall. Figure 2 shows some example logical forms produced by ASP in the evaluation.

The annotated sample of logical forms allows us to estimate precision for each system as a function of the number of correct extractions (Figure 3). The number of correct extractions is directly proportional to recall, and was estimated from the total number of extractions and precision at each rank in the sample. All three systems initially have high precision, implying that their extracted logical forms express facts found in the sentence. However, ASP produces 3 times more correct logical forms than either baseline because it jointly analyzes syntax and semantics. The baselines suffer from reduced recall because they depend on receiving an accurate syntactic parse as input; syntactic parsing errors cause these systems to fail.

Examining the incorrect logical forms produced by ASP reveals that incorrect mention detection is by far the most common source of mistakes. Approximately 50% of errors are caused by marking common nouns as entity mentions (e.g., marking "coin" as a COMPANY). These errors occur because the knowledge base contains many infrequently mentioned entities with relatively common names. Another 30% of errors are caused by assigning an incorrect type to a common proper noun (e.g, marking "Bolivia" as a CITY). This analysis suggests that performing entity linking before parsing could significantly reduce errors.

Sentence: *De Niro and Joe Pesci in "Goodfellas"* offered a virtuoso display of the director's bravura cinematic technique and reestablished, enhanced, and consolidated his reputation.
Annotation:
LF: $\lambda x.\forall p \in \{\lambda d.\text{M}(d, \text{“de niro”}), \lambda j.\text{M}(j, \text{“joe pesci”})\} \exists y.p(x) \wedge \text{STARREDINMOVIE}(x, y) \wedge \text{M}(y, \text{“goodfellas”})$
Instances: STARREDINMOVIE(de niro, goodfellas), STARREDINMOVIE(joe pesci, goodfellas)
Prediction:
LF: $\lambda x.\forall p \in \{\lambda d.\text{M}(d, \text{“de niro”}), \lambda j.\text{M}(j, \text{“joe pesci”})\} \exists y.p(x) \wedge \text{STARREDINMOVIE}(x, y) \wedge \text{M}(y, \text{“goodfellas”})$
Instances: STARREDINMOVIE(de niro, goodfellas), STARREDINMOVIE(joe pesci, goodfellas)
Logical form accuracy: 1 / 1 **Extraction Precision:** 2 / 2 **Extraction Recall:** 2 / 2

Sentence: In addition to the University of Illinois, *Champaign is also home to Parkland College*.
Annotation:
LF: $\exists c, p.\text{M}(c, \text{“champaign”}) \wedge \text{CITY}(c) \wedge \text{M}(p, \text{“parkland college”}) \wedge \text{UNIVERSITYINCITY}(p, c)$
Instances: CITY(champaign), UNIVERSITYINCITY(parkland college, champaign)
Prediction:
LF 1: $\lambda x.\exists y\text{M}(y, \text{“illinois”}) \wedge \text{M}(x, \text{“university”}) \wedge \text{CITYLOCATEDINSTATE}(x, y)$
LF 2: $\exists c, p.\text{M}(c, \text{“champaign”}) \wedge \text{CITY}(c) \wedge \text{M}(p, \text{“parkland college”}) \wedge \text{UNIVERSITYINCITY}(p, c)$
Instances: CITY(champaign), UNIVERSITYINCITY(parkland college, champaign),
CITYLOCATEDINSTATE(university, illinois)
Logical form accuracy: 1 / 1 **Extraction Precision:** 2 / 3 **Extraction Recall:** 2 / 2

Figure 4: Two test examples with ASP's predictions and error calculations. The annotated logical forms are for the italicized sentence spans, while the extracted logical forms are for the underlined spans.

5.5.3 Annotated Sentence Evaluation

A limitation of the previous evaluation is that it does not measure the completeness of predicted logical forms, nor estimate what portion of sentences are left unanalyzed. We conducted a second evaluation to measure these quantities.

The data for this evaluation consists of sentences annotated with logical forms for subspans. We manually annotated Wikipedia sentences from the held-out set with logical forms for the largest subspans for which a logical form existed. To avoid trivial cases, we only annotated logical forms containing at least one category or relation predicate and at least one mention. We also chose not to annotate mentions of entities that are not in the knowledge base, as no system would be able to correctly identify them. The corpus contains 97 sentences with 100 annotated logical forms.

We measured performance using two metrics: logical form accuracy, and extraction precision/recall. Logical form accuracy examines the predicted logical form for the smallest subspan of the sentence containing the annotated span, and marks this prediction correct if it exactly matches the annotation. A limitation of this metric is that it does not assign partial credit to logical forms that are close to, but do not exactly match, the annotation. The extraction metric assigns partial credit by computing the precision and recall of the category and relation instances entailed by the predicted logical form, using those entailed by the annotated logical form as the gold standard. Figure 4 shows the computation of both error metrics on two examples from the test corpus.

Table 3 shows the results of the annotated sentence evaluation. ASP outperforms both baselines in logical form accuracy and extraction recall, suggesting that it produces more complete analyses than either baseline. The extraction precision of 90% suggests that ASP rarely extracts incorrect information. Precision is higher in this evaluation because every sentence in the data set has at least one correct extraction.

6 Discussion

We present an approach to training a joint syntactic and semantic parser. Our parser ASP produces a full syntactic parse of any sentence, while simultaneously producing logical forms for sentence spans that have a semantic representation within its predicate vocabulary. The parser is trained by jointly optimizing performance on a syntactic parsing task and a distantly-supervised relation extraction task. Experimental results demonstrate that jointly analyzing syntax and semantics triples the number of extracted logical forms over approaches that first analyze syntax, then semantics. However, we also find that incorporating semantics slightly reduces syntactic parsing performance. Poor entity mention detection is a major source of error in both cases, suggesting that future work should consider integrating entity linking with joint syntactic and semantic parsing.

Acknowledgments

This work was supported in part by DARPA under award FA8750-13-2-0005. We additionally thank Jamie Callan and Chris Ré's Hazy group for collecting and processing the Wikipedia corpus.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of the 17th International Conference on Computational Linguistics - Volume 1*.

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: an approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Johan Bos. 2005. Towards wide-coverage semantic interpretation. In *In Proceedings of Sixth International Workshop on Computational Semantics IWCS-6*.

Qingqing Cai and Alexander Yates. 2013a. Large-scale Semantic Parsing via Schema Matching and Lexicon Extension. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Qingqing Cai and Alexander Yates. 2013b. Semantic Parsing Freebase: Towards Open-domain Semantic Parsing. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics (*SEM)*.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*.

Stephen Clark and James R. Curran. 2004. The importance of supertagging for wide-coverage CCG parsing. In *Proceedings of the 20th International Conference on Computational Linguistics*.

Stephen Clark and James R. Curran. 2007a. Perceptron training for a wide-coverage lexicalized-grammar parser. In *Proceedings of the Workshop on Deep Linguistic Processing*.

Stephen Clark and James R. Curran. 2007b. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*.

Ruifang Ge and Raymond J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*.

Julia Hockenmaier and Mark Steedman. 2002a. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of Third International Conference on Language Resources and Evaluation*.

Julia Hockenmaier and Mark Steedman. 2002b. Generative models for statistical parsing with combinatory categorial grammar. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*.

Julia Hockenmaier. 2003a. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Julia Hockenmaier. 2003b. Parsing with generative models of predicate-argument structure. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke S. Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Rohit J. Kate and Raymond J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*.

Jayant Krishnamurthy and Tom M. Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Mike Lewis and Mark Steedman. 2013. Combined distributional and logical semantics. *Transactions of the Association for Computational Linguistics*, 1:179–192.

Percy Liang, Michael I. Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the Association for Computational Linguistics*, Portland, Oregon. Association for Computational Linguistics.

Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Nathan D. Ratliff, J. Andrew Bagnell, and Martin A. Zinkevich. 2006. (online) subgradient methods for structured prediction. *Artificial Intelligence and Statistics*.

Mark Steedman. 1996. *Surface Structure and Interpretation*. The MIT Press, Cambridge, MA, USA.

Yuk Wah Wong and Raymond J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL*.

Yuk Wah Wong and Raymond J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, Maya Ramanath, Volker Tresp, and Gerhard Weikum. 2012. Natural language questions for the web of data. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the thirteenth national conference on Artificial Intelligence*.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence*.

Learning Semantic Hierarchies via Word Embeddings

Ruiji Fu[†], **Jiang Guo**[†], **Bing Qin**[†], **Wanxiang Che**[†], **Haifeng Wang**[‡], **Ting Liu**[†*]
[†]Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
[‡]Baidu Inc., Beijing, China
{rjfu, jguo, bqin, car, tliu}@ir.hit.edu.cn
wanghaifeng@baidu.com

Abstract

Semantic hierarchy construction aims to build structures of concepts linked by hypernym–hyponym ("is-a") relations. A major challenge for this task is the automatic discovery of such relations. This paper proposes a novel and effective method for the construction of semantic hierarchies based on word embeddings, which can be used to measure the semantic relationship between words. We identify whether a candidate word pair has hypernym–hyponym relation by using the word-embedding-based semantic projections between words and their hypernyms. Our result, an F-score of 73.74%, outperforms the state-of-the-art methods on a manually labeled test dataset. Moreover, combining our method with a previous manually-built hierarchy extension method can further improve F-score to 80.29%.

1 Introduction

Semantic hierarchies are natural ways to organize knowledge. They are the main components of ontologies or semantic thesauri (Miller, 1995; Suchanek et al., 2008). In the WordNet hierarchy, senses are organized according to the "is-a" relations. For example, "dog" and "canine" are connected by a directed edge. Here, "canine" is called a hypernym of "dog." Conversely, "dog" is a hyponym of "canine." As key sources of knowledge, semantic thesauri and ontologies can support many natural language processing applications. However, these semantic resources are limited in its scope and domain, and their manual construction is knowledge intensive and time consuming. Therefore, many researchers have attempted to automatically extract semantic relations or to construct taxonomies.

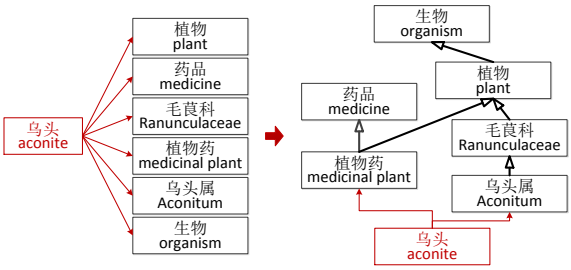


Figure 1: An example of semantic hierarchy construction.

A major challenge for this task is the automatic discovery of hypernym-hyponym relations. Fu et al. (2013) propose a distant supervision method to extract hypernyms for entities from multiple sources. The output of their model is a list of hypernyms for a given enity (left panel, Figure 1). However, there usually also exists hypernym–hyponym relations among these hypernyms. For instance, "植物 (plant)" and "毛茛科 (Ranunculaceae)" are both hypernyms of the entity "乌头 (aconit)," and "植物 (plant)" is also a hypernym of "毛茛科 (Ranunculaceae)." Given a list of hypernyms of an entity, our goal in the present work is to construct a semantic hierarchy of these hypernyms (right panel, Figure 1).[1]

Some previous works extend and refine manually-built semantic hierarchies by using other resources (e.g., Wikipedia) (Suchanek et al., 2008). However, the coverage is limited by the scope of the resources. Several other works relied heavily on lexical patterns, which would suffer from deficiency because such patterns can only cover a small proportion of complex linguistic circumstances (Hearst, 1992; Snow et al., 2005).

*Email correspondence.

[1]In this study, we focus on Chinese semantic hierarchy construction. The proposed method can be easily adapted to other languages.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1199–1209,
Baltimore, Maryland, USA, June 23-25 2014.

Besides, distributional similarity methods (Kotlerman et al., 2010; Lenci and Benotto, 2012) are based on the assumption that a term can only be used in contexts where its hypernyms can be used and that a term might be used in any contexts where its hyponyms are used. However, it is not always rational. Our previous method based on web mining (Fu et al., 2013) works well for hypernym extraction of entity names, but it is unsuitable for semantic hierarchy construction which involves many words with broad semantics. Moreover, all of these methods do not use the word semantics effectively.

This paper proposes a novel approach for semantic hierarchy construction based on word embeddings. Word embeddings, also known as distributed word representations, typically represent words with dense, low-dimensional and real-valued vectors. Word embeddings have been empirically shown to preserve linguistic regularities, such as the semantic relationship between words (Mikolov et al., 2013b). For example, $v(\texttt{king}) - v(\texttt{queen}) \approx v(\texttt{man}) - v(\texttt{woman})$, where $v(w)$ is the embedding of the word w. We observe that a similar property also applies to the hypernym–hyponym relationship (Section 3.3), which is the main inspiration of the present study.

However, we further observe that hypernym–hyponym relations are more complicated than a single offset can represent. To address this challenge, we propose a more sophisticated and general method — learning a linear projection which maps words to their hypernyms (Section 3.3.1). Furthermore, we propose a piecewise linear projection method based on relation clustering to better model hypernym–hyponym relations (Section 3.3.2). Subsequently, we identify whether an unknown word pair is a hypernym–hyponym relation using the projections (Section 3.4). To the best of our knowledge, we are the first to apply word embeddings to this task.

For evaluation, we manually annotate a dataset containing 418 Chinese entities and their hypernym hierarchies, which is the first dataset for this task as far as we know. The experimental results show that our method achieves an F-score of 73.74% which significantly outperforms the previous state-of-the-art methods. Moreover, combining our method with the manually-built hierarchy extension method proposed by Suchanek et al. (2008) can further improve F-score to 80.29%.

2 Background

As main components of ontologies, semantic hierarchies have been studied by many researchers. Some have established concept hierarchies based on manually-built semantic resources such as WordNet (Miller, 1995). Such hierarchies have good structures and high accuracy, but their coverage is limited to fine-grained concepts (e.g., "Ranunculaceae" is not included in WordNet.). We have made similar obsevation that about a half of hypernym–hyponym relations are absent in a Chinese semantic thesaurus. Therefore, a broader range of resources is needed to supplement the manually built resources. In the construction of the famous ontology YAGO, Suchanek et al. (2008) link the categories in Wikipedia onto WordNet. However, the coverage is still limited by the scope of Wikipedia.

Several other methods are based on lexical patterns. They use manually or automatically constructed lexical patterns to mine hypernym–hyponym relations from text corpora. A hierarchy can then be built based on these pairwise relations. The pioneer work by Hearst (1992) has found out that linking two noun phrases (NPs) via certain lexical constructions often implies hypernym relations. For example, NP_1 is a hypernym of NP_2 in the lexical pattern "such NP_1 as NP_2." Snow et al. (2005) propose to automatically extract large numbers of lexico-syntactic patterns and subsequently detect hypernym relations from a large newswire corpus. Their method relies on accurate syntactic parsers, and the quality of the automatically extracted patterns is difficult to guarantee. Generally speaking, these pattern-based methods often suffer from low recall or precision because of the coverage or the quality of the patterns.

The distributional methods assume that the contexts of hypernyms are broader than the ones of their hyponyms. For distributional similarity computing, each word is represented as a semantic vector composed of the pointwise mutual information (PMI) with its contexts. Kotlerman et al. (2010) design a directional distributional measure to infer hypernym–hyponym relations based on the standard IR Average Precision evaluation measure. Lenci and Benotto (2012) propose another measure focusing on the contexts that hypernyms do not share with their hyponyms. However, broader semantics may not always infer broader contexts. For example, for terms "Obama' and

"American people", it is hard to say whose contexts are broader.

Our previous work (Fu et al., 2013) applies a web mining method to discover the hypernyms of Chinese entities from multiple sources. We assume that the hypernyms of an entity co-occur with it frequently. It works well for named entities. But for class names (e.g., singers in Hong Kong, tropical fruits) with wider range of meanings, this assumption may fail.

In this paper, we aim to identify hypernym–hyponym relations using word embeddings, which have been shown to preserve good properties for capturing semantic relationship between words.

3 Method

In this section, we first define the task formally. Then we elaborate on our proposed method composed of three major steps, namely, word embedding training, projection learning, and hypernym–hyponym relation identification.

3.1 Task Definition

Given a list of hypernyms of an entity, our goal is to construct a semantic hierarchy on it (Figure 1). We represent the hierarchy as a directed graph G, in which the nodes denote the words, and the edges denote the hypernym–hyponym relations. Hypernym-hyponym relations are *asymmetric* and *transitive* when words are unambiguous:

- $\forall x, y \in L : x \xrightarrow{H} y \Rightarrow \neg(y \xrightarrow{H} x)$
- $\forall x, y, z \in L : (x \xrightarrow{H} z \wedge z \xrightarrow{H} y) \Rightarrow x \xrightarrow{H} y$

Here, L denotes the list of hypernyms. x, y and z denote the hypernyms in L. We use $\xrightarrow{H}$ to represent a hypernym–hyponym relation in this paper. Actually, x, y and z are unambiguous as the hypernyms of a certain entity. Therefore, G should be a directed acyclic graph (DAG).

3.2 Word Embedding Training

Various models for learning word embeddings have been proposed, including neural net language models (Bengio et al., 2003; Mnih and Hinton, 2008; Mikolov et al., 2013b) and spectral models (Dhillon et al., 2011). More recently, Mikolov et al. (2013a) propose two log-linear models, namely the *Skip-gram* and *CBOW* model, to efficiently induce word embeddings. These two models can be trained very efficiently on a large-scale corpus because of their low time complexity.

No.	Examples
1	v(虾) − v(对虾) ≈ v(鱼) − v(金鱼) v(shrimp) − v(prawn) ≈ v(fish) − v(gold fish)
2	v(工人) − v(木匠) ≈ v(演员) − v(小丑) v(laborer) − v(carpenter) ≈ v(actor) − v(clown)
3	v(工人) − v(木匠) $\not\approx$ v(鱼) − v(金鱼) v(laborer) − v(carpenter) $\not\approx$ v(fish) − v(gold fish)

Table 1: Embedding offsets on a sample of hypernym–hyponym word pairs.

Additionally, their experiment results have shown that the *Skip-gram* model performs best in identifying semantic relationship among words. Therefore, we employ the *Skip-gram* model for estimating word embeddings in this study.

The *Skip-gram* model adopts log-linear classifiers to predict context words given the current word $\mathbf{w}(t)$ as input. First, $\mathbf{w}(t)$ is projected to its embedding. Then, log-linear classifiers are employed, taking the embedding as input and predict $\mathbf{w}(t)$'s context words within a certain range, e.g. k words in the left and k words in the right. After maximizing the log-likelihood over the entire dataset using stochastic gradient descent (SGD), the embeddings are learned.

3.3 Projection Learning

Mikolov et al. (2013b) observe that word embeddings preserve interesting linguistic regularities, capturing a considerable amount of syntactic/semantic relations. Looking at the well-known example: $v(\texttt{king}) - v(\texttt{queen}) \approx v(\texttt{man}) - v(\texttt{woman})$, it indicates that the embedding offsets indeed represent the shared semantic relation between the two word pairs.

We observe that the same property also applies to some hypernym–hyponym relations. As a preliminary experiment, we compute the embedding offsets between some randomly sampled hypernym–hyponym word pairs and measure their similarities. The results are shown in Table 1.

The first two examples imply that a word can also be mapped to its hypernym by utilizing word embedding offsets. However, the offset from "carpenter" to "laborer" is distant from the one from "gold fish" to "fish," indicating that hypernym–hyponym relations should be more complicated than a single vector offset can represent. To verify this hypothesis, we compute the embedding offsets over all hypernym–

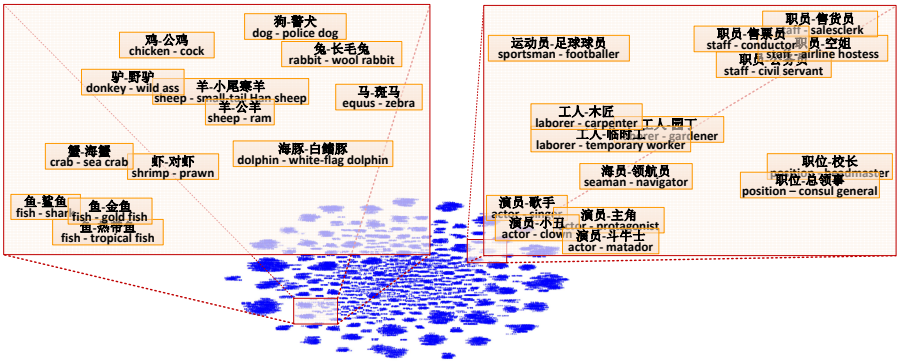


Figure 2: Clusters of the vector offsets in training data. The figure shows that the vector offsets distribute in some clusters. The left cluster shows some hypernym–hyponym relations about animals. The right one shows some relations about people's occupations.

hyponym word pairs in our training data and visualize them.[2] Figure 2 shows that the relations are adequately distributed in the clusters, which implies that hypernym–hyponym relations indeed can be decomposed into more fine-grained relations. Moreover, the relations about animals are spatially close, but separate from the relations about people's occupations.

To address this challenge, we propose to learn the hypernym–hyponym relations using projection matrices.

3.3.1 A Uniform Linear Projection

Intuitively, we assume that all words can be projected to their hypernyms based on a uniform transition matrix. That is, given a word x and its hypernym y, there exists a matrix Φ so that $y = \Phi x$. For simplicity, we use the same symbols as the words to represent the embedding vectors. Obtaining a consistent exact Φ for the projection of all hypernym–hyponym pairs is difficult. Instead, we can learn an approximate Φ using Equation 1 on the training data, which minimizes the mean-squared error:

$$\Phi^* = \arg\min_{\Phi} \frac{1}{N} \sum_{(x,y)} \| \Phi x - y \|^2 \qquad (1)$$

where N is the number of (x, y) word pairs in the training data. This is a typical linear regression problem. The only difference is that our predictions are multi-dimensional vectors instead of scalar values. We use SGD for optimization.

[2]Principal Component Analysis (PCA) is applied for dimensionality reduction.

3.3.2 Piecewise Linear Projections

A uniform linear projection may still be underrepresentative for fitting all of the hypernym–hyponym word pairs, because the relations are rather diverse, as shown in Figure 2. To better model the various kinds of hypernym–hyponym relations, we apply the idea of piecewise linear regression (Ritzema, 1994) in this study.

Specifically, the input space is first segmented into several regions. That is, all word pairs (x, y) in the training data are first clustered into several groups, where word pairs in each group are expected to exhibit similar hypernym–hyponym relations. Each word pair (x, y) is represented with their vector offsets: $y - x$ for clustering. The reasons are twofold: (1) Mikolov's work has shown that the vector offsets imply a certain level of semantic relationship. (2) The vector offsets distribute in clusters well, and the word pairs which are close indeed represent similar relations, as shown in Figure 2.

Then we learn a separate projection for each cluster, respectively (Equation 2).

$$\Phi_k^* = \arg\min_{\Phi_k} \frac{1}{N_k} \sum_{(x,y)\in C_k} \| \Phi_k x - y \|^2 \qquad (2)$$

where N_k is the amount of word pairs in the k^{th} cluster C_k.

We use the k-means algorithm for clustering, where k is tuned on a development dataset.

3.3.3 Training Data

To learn the projection matrices, we extract training data from a Chinese semantic thesaurus, Tongyi Cilin (Extended) (CilinE for short) which

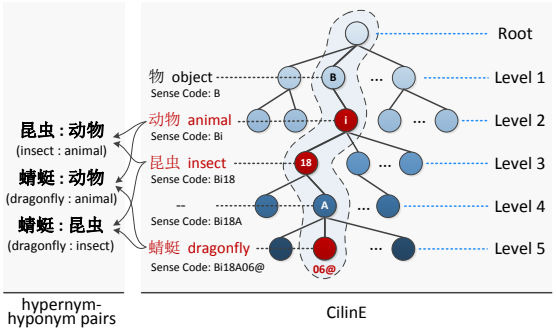


Figure 3: Hierarchy of CilinE and an Example of Training Data Generation

contains 100,093 words (Che et al., 2010).[3] CilinE is organized as a hierarchy of five levels, in which the words are linked by hypernym–hyponym relations (right panel, Figure 3). Each word in CilinE has one or more sense codes (some words are polysemous) that indicate its position in the hierarchy.

The senses of words in the first level, such as "物 (`object`)" and "时间 (`time`)," are very general. The fourth level only has sense codes without real words. Therefore, we extract words in the second, third and fifth levels to constitute hypernym–hyponym pairs (left panel, Figure 3).

Note that mapping one hyponym to multiple hypernyms with the same projection (Φx is unique) is difficult. Therefore, the pairs with the same hyponym but different hypernyms are expected to be clustered into separate groups. Figure 3 shows that the word "`dragonfly`" in the fifth level has two hypernyms: "`insect`" in the third level and "`animal`" in the second level. Hence the relations `dragonfly` $\xrightarrow{H}$ `insect` and `dragonfly` $\xrightarrow{H}$ `animal` should fall into different clusters.

In our implementation, we apply this constraint by simply dividing the training data into two categories, namely, *direct* and *indirect*. Hypernym-hyponym word pair (x, y) is classified into the *direct* category, only if there doesn't exist another word z in the training data, which is a hypernym of x and a hyponym of y. Otherwise, (x, y) is classified into the *indirect* category. Then, data in these two categories are clustered separately.

[3] `www.ltp-cloud.com/download/`

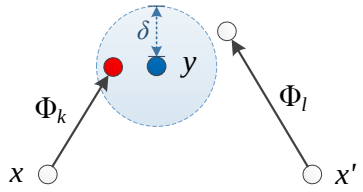


Figure 4: In this example, $\Phi_k x$ is located in the circle with center y and radius δ. So y is considered as a hypernym of x. Conversely, y is not a hypernym of x'.

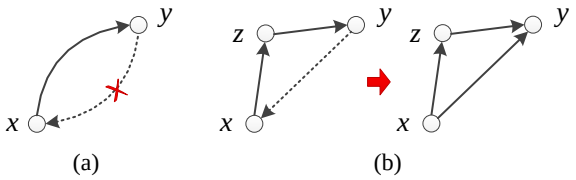


Figure 5: (a) If $d(\Phi_j y, x) > d(\Phi_k x, y)$, we remove the path from y to x; (b) if $d(\Phi_j y, x) > d(\Phi_k x, z)$ and $d(\Phi_j y, x) > d(\Phi_i z, y)$, we reverse the path from y to x.

3.4 Hypernym-hyponym Relation Identification

Upon obtaining the clusters of training data and the corresponding projections, we can identify whether two words have a hypernym–hyponym relation. Given two words x and y, we find cluster C_k whose center is closest to the offset $y - x$, and obtain the corresponding projection Φ_k. For y to be considered a hypernym of x, one of the two conditions below must hold.

Condition 1: The projection Φ_k puts $\Phi_k x$ close enough to y (Figure 4). Formally, the euclidean distance between $\Phi_k x$ and y: $d(\Phi_k x, y)$ must be less than a threshold δ.

$$d(\Phi_k x, y) = \| \Phi_k x - y \|^2 < \delta \qquad (3)$$

Condition 2: There exists another word z satisfying $x \xrightarrow{H} z$ and $z \xrightarrow{H} y$. In this case, we use the transitivity of hypernym–hyponym relations.

Besides, the final hierarchy should be a DAG as discussed in Section 3.1. However, the projection method cannot guarantee that theoretically, because the projections are learned from pairwise hypernym–hyponym relations without the whole hierarchy structure. All pairwise hypernym–hyponym relation identification methods would suffer from this problem actually. It is an interesting problem how to construct a globally opti-

mal semantic hierarchy conforming to the form of a DAG. But this is not the focus of this paper. So if some conflicts occur, that is, a relation circle exists, we remove or reverse the weakest path heuristically (Figure 5). If a circle has only two nodes, we remove the weakest path. If a circle has more than two nodes, we reverse the weakest path to form an *indirect* hypernym–hyponym relation.

4 Experimental Setup

4.1 Experimental Data

In this work, we learn word embeddings from a Chinese encyclopedia corpus named Baidubaike[4], which contains about 30 million sentences (about 780 million words). The Chinese segmentation is provided by the open-source Chinese language processing platform LTP[5] (Che et al., 2010). Then, we employ the *Skip-gram* method (Section 3.2) to train word embeddings. Finally we obtain the embedding vectors of 0.56 million words.

The training data for projection learning is collected from CilinE (Section 3.3.3). We obtain 15,247 word pairs of hypernym–hyponym relations (9,288 for *direct* relations and 5,959 for *indirect* relations).

For evaluation, we collect the hypernyms for 418 entities, which are selected randomly from Baidubaike, following Fu et al. (2013). We then ask two annotators to manually label the semantic hierarchies of the correct hypernyms. The final data set contains 655 unique hypernyms and 1,391 hypernym–hyponym relations among them. We randomly split the labeled data into 1/5 for development and 4/5 for testing (Table 2). The hierarchies are represented as relations of pairwise words. We measure the inter-annotator agreement using the kappa coefficient (Siegel and Castellan Jr, 1988). The kappa value is 0.96, which indicates a good strength of agreement.

4.2 Evaluation Metrics

We use precision, recall, and F-score as our metrics to evaluate the performances of the methods.

Since hypernym–hyponym relations and its reverse (hyponym–hypernym) have one-to-one correspondence, their performances are equal. For simplicity, we only report the performance of the former in the experiments.

[4]Baidubaike (baike.baidu.com) is one of the largest Chinese encyclopedias containing more than 7.05 million entries as of September, 2013.

[5]www.ltp-cloud.com/demo/

Relation	# of word pairs	
	Dev.	Test
hypernym–hyponym	312	1,079
hyponym–hypernym*	312	1,079
unrelated	1,044	3,250
Total	1,668	5,408

Table 2: The evaluation data. *Since hypernym–hyponym relations and hyponym–hypernym relations have one-to-one correspondence, their numbers are the same.

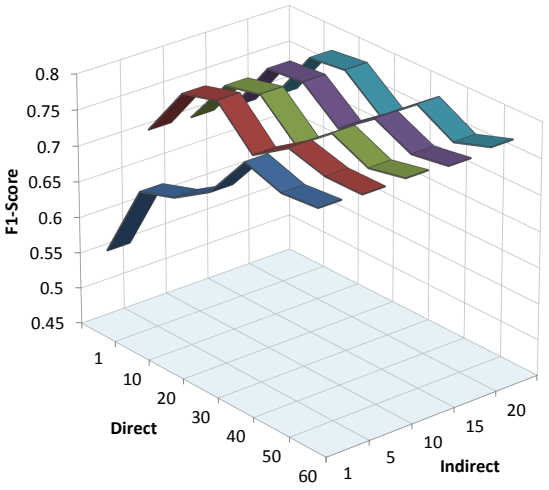


Figure 6: Performance on development data w.r.t. cluster size.

5 Results and Analysis

5.1 Varying the Amount of Clusters

We first evaluate the effect of different number of clusters based on the development data. We vary the numbers of the clusters both for the *direct* and *indirect* training word pairs.

As shown in Figure 6, the performance of clustering is better than non-clustering (when the cluster number is 1), thus providing evidences that learning piecewise projections based on clustering is reasonable. We finally set the numbers of the clusters of *direct* and *indirect* to 20 and 5, respectively, where the best performances are achieved on the development data.

5.2 Comparison with Previous Work

In this section, we compare the proposed method with previous methods, including manually-built hierarchy extension, pairwise relation extraction

	P(%)	R(%)	F(%)
$M_{Wiki+CilinE}$	92.41	60.61	73.20
$M_{Pattern}$	97.47	21.41	35.11
M_{Snow}	60.88	25.67	36.11
$M_{balApinc}$	54.96	53.38	54.16
M_{invCL}	49.63	62.84	55.46
M_{Fu}	87.40	48.19	62.13
M_{Emb}	80.54	67.99	**73.74**
$M_{Emb+CilinE}$	80.59	72.42	76.29
$M_{Emb+Wiki+CilinE}$	79.78	80.81	**80.29**

Table 3: Comparison of the proposed method with existing methods in the test set.

Pattern	Translation
w 是[一个|一种] h	w is a [a kind of] h
w [、] 等 h	w[,] and other h
h [，] 叫[做] w	h[,] called w
h [，] [像]如 w	h[,] such as w
h [，] 特别是 w	h[,] especially w

Table 4: Chinese Hearst-style lexical patterns. The contents in square brackets are omissible.

based on patterns, word distributions, and web mining (Section 2). Results are shown in Table 3.

5.2.1 Overall Comparison

$M_{Wiki+CilinE}$ refers to the manually-built hierarchy extension method of Suchanek et al. (2008). In our experiment, we use the category taxonomy of Chinese Wikipedia[6] to extend CilinE. Table 3 shows that this method achieves a high precision but also a low recall, mainly because of the limited scope of Wikipedia.

$M_{Pattern}$ refers to the pattern-based method of Hearst (1992). We extract hypernym–hyponym relations in the Baidubaike corpus, which is also used to train word embeddings (Section 4.1). We use the Chinese Hearst-style patterns (Table 4) proposed by Fu et al. (2013), in which w represents a word, and h represents one of its hypernyms. The result shows that only a small part of the hypernyms can be extracted based on these patterns because only a few hypernym relations are expressed in these fixed patterns, and many are expressed in highly flexible manners.

In the same corpus, we apply the method M_{Snow} originally proposed by Snow et al. (2005). The same training data for projections learning from CilinE (Section 3.3.3) is used as seed hypernym–hyponym pairs. Lexico-syntactic patterns are extracted from the Baidubaike corpus by using the seeds. We then develop a logistic regression classifier based on the patterns to recognize hypernym–hyponym relations. This method relies on an accurate syntactic parser, and the quality of the automatically extracted patterns is difficult to guarantee.

We re-implement two previous distributional methods $M_{balApinc}$ (Kotlerman et al., 2010) and M_{invCL} (Lenci and Benotto, 2012) in the Baidubaike corpus. Each word is represented as a feature vector in which each dimension is the PMI value of the word and its context words. We compute a score for each word pair and apply a threshold to identify whether it is a hypernym–hyponym relation.

M_{Fu} refers to our previous web mining method (Fu et al., 2013). This method mines hypernyms of a given word w from multiple sources and returns a ranked list of the hypernyms. We select the hypernyms with scores over a threshold of each word in the test set for evaluation. This method assumes that frequent co-occurrence of a noun or noun phrase n in multiple sources with w indicate possibility of n being a hypernym of w. The results presented in Fu et al. (2013) show that the method works well when w is an entity, but not when w is a word with a common semantic concept. The main reason may be that there are relatively more introductory pages about entities than about common words in the Web.

M_{Emb} is the proposed method based on word embeddings. Table 3 shows that the proposed method achieves a better recall and F-score than all of the previous methods do. It can significantly ($p < 0.01$) improve the F-score over the state-of-the-art method $M_{Wiki+CilinE}$.

M_{Emb} and M_{CilinE} can also be combined. The combination strategy is to simply merge all positive results from the two methods together, and then to infer new relations based on the transitivity of hypernym–hyponym relations. The F-score is further improved from 73.74% to 76.29%. Note that, the combined method achieves a 4.43% recall improvement over M_{Emb}, but the precision is almost unchanged. The reason is that the inference based on the relations identified automatically may lead to error propagation. For example, the relation $x \xrightarrow{H} y$ is incorrectly identified by M_{Emb}.

[6] dumps.wikimedia.org/zhwiki/20131205/

	P(%)	R(%)	F(%)
$M_{Wiki+CilinE}$	80.39	19.29	31.12
$M_{Emb+CilinE}$	71.16	52.80	60.62
$M_{Emb+Wiki+CilinE}$	69.13	61.65	**65.17**

Table 5: Performance on the out-of-CilinE data in the test set.

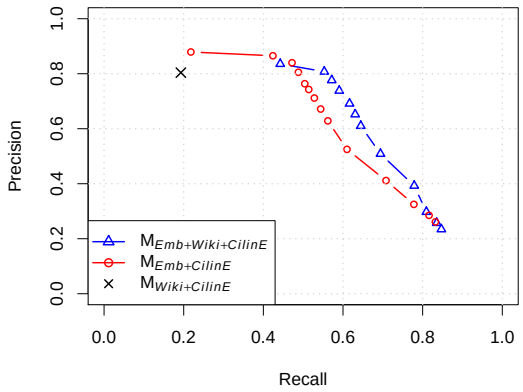


Figure 7: Precision-Recall curves on the out-of-CilinE data in the test set.

When the relation $y \xrightarrow{H} z$ from M_{CilinE} is added, it will cause a new incorrect relation $x \xrightarrow{H} z$.

Combining M_{Emb} with $M_{Wiki+CilinE}$ achieves a 7% F-score improvement over the best baseline $M_{Wiki+CilinE}$. Therefore, the proposed method is complementary to the manually-built hierarchy extension method (Suchanek et al., 2008).

5.2.2 Comparison on the Out-of-CilinE Data

We are greatly interested in the practical performance of the proposed method on the hypernym–hyponym relations outside of CilinE. We say a word pair is outside of CilinE, as long as there is one word in the pair not existing in CilinE. In our test data, about 62% word pairs are outside of CilinE. Table 5 shows the performances of the best baseline method and our method on the out-of-CilinE data. The method exploiting the taxonomy in Wikipedia, $M_{Wiki+CilinE}$, achieves the highest precision but has a low recall. By contrast, our method can discover more hypernym–hyponym relations with some loss of precision, thereby achieving a more than 29% F-score improvement. The combination of these two methods achieves a further 4.5% F-score improvement over $M_{Emb+CilinE}$. Generally speaking, the proposed method greatly improves the recall but damages the precision.

Actually, we can get different precisions and recalls by adjusting the threshold δ (Equation 3). Figure 7 shows that $M_{Emb+CilinE}$ achieves a higher precision than $M_{Wiki+CilinE}$ when their recalls are the same. When they achieve the same precision, the recall of $M_{Emb+CilinE}$ is higher.

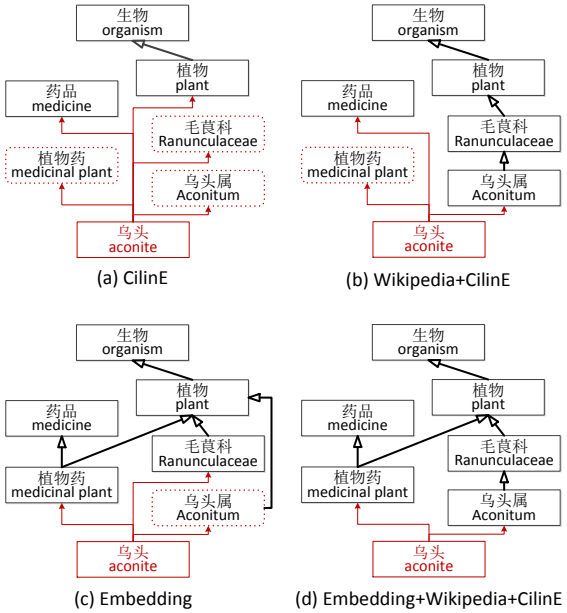


Figure 8: An example for error analysis. The red paths refer to the relations between the named entity and its hypernyms extracted using the web mining method (Fu et al., 2013). The black paths with hollow arrows denote the relations identified by the different methods. The boxes with dotted borders refer to the concepts which are not linked to correct positions.

5.3 Error Analysis and Discussion

We analyze error cases after experiments. Some cases are shown in Figure 8. We can see that there is only one general relation "植物 (plant)" $\xrightarrow{H}$ "生物 (organism)" existing in CilinE. Some fine-grained relations exist in Wikipedia, but the coverage is limited. Our method based on word embeddings can discover more hypernym–hyponym relations than the previous methods can. When we combine the methods together, we get the correct hierarchy.

Figure 8 shows that our method loses the relation "乌头属 (Aconitum)" $\xrightarrow{H}$ "毛茛科 (Ranunculaceae)." It is because they are very semantically similar (their cosine similarity is 0.9038). Their representations are so close to each other in the embedding space that we have not find projections suitable for these pairs. The

error statistics show that when the cosine similarities of word pairs are greater than 0.8, the recall is only 9.5%. This kind of error accounted for about 10.9% among all the errors in our test set. One possible solution may be adding more data of this kind to the training set.

6 Related Work

In addition to the works mentioned in Section 2, we introduce another set of related studies in this section.

Evans (2004), Ortega-Mendoza et al. (2007), and Sang (2007) consider web data as a large corpus and use search engines to identify hypernyms based on the lexical patterns of Hearst (1992). However, the low quality of the sentences in the search results negatively influence the precision of hypernym extraction.

Following the method for discovering patterns automatically (Snow et al., 2005), McNamee et al. (2008) apply the same method to extract hypernyms of entities in order to improve the performance of a question answering system. Ritter et al. (2009) propose a method based on patterns to find hypernyms on arbitrary noun phrases. They use a support vector machine classifier to identify the correct hypernyms from the candidates that match the patterns. As our experiments show, pattern-based methods suffer from low recall because of the low coverage of patterns.

Besides Kotlerman et al. (2010) and Lenci and Benotto (2012), other researchers also propose directional distributional similarity methods (Weeds et al., 2004; Geffet and Dagan, 2005; Bhagat et al., 2007; Szpektor et al., 2007; Clarke, 2009). However, their basic assumption that a hyponym can only be used in contexts where its hypernyms can be used and that a hypernym might be used in all of the contexts where its hyponyms are used may not always rational.

Snow et al. (2006) provides a global optimization scheme for extending WordNet, which is different from the above-mentioned pairwise relationships identification methods.

Word embeddings have been successfully applied in many applications, such as in sentiment analysis (Socher et al., 2011b), paraphrase detection (Socher et al., 2011a), chunking, and named entity recognition (Turian et al., 2010; Collobert et al., 2011). These applications mainly utilize the representing power of word embeddings to alleviate the problem of data sparsity. Mikolov et al. (2013a) and Mikolov et al. (2013b) further observe that the semantic relationship of words can be induced by performing simple algebraic operations with word vectors. Their work indicates that word embeddings preserve some interesting linguistic regularities, which might provide support for many applications. In this paper, we improve on their work by learning multiple linear projections in the embedding space, to model hypernym–hyponym relationships within different clusters.

7 Conclusion and Future Work

This paper proposes a novel method for semantic hierarchy construction based on word embeddings, which are trained using a large-scale corpus. Using the word embeddings, we learn the hypernym–hyponym relationship by estimating projection matrices which map words to their hypernyms. Further improvements are made using a cluster-based approach in order to model the more fine-grained relations. Then we propose a few simple criteria to identity whether a new word pair is a hypernym–hyponym relation. Based on the pairwise hypernym–hyponym relations, we build semantic hierarchies automatically.

In our experiments, the proposed method significantly outperforms state-of-the-art methods and achieves the best F1-score of 73.74% on a manually labeled test dataset. Further experiments show that our method is complementary to the previous manually-built hierarchy extension methods.

For future work, we aim to improve word embedding learning under the guidance of hypernym–hyponym relations. By including the hypernym–hyponym relation constraints while training word embeddings, we expect to improve the embeddings such that they become more suitable for this task.

Acknowledgments

This work was supported by National Natural Science Foundation of China (NSFC) via grant 61133012, 61273321 and the National 863 Leading Technology Research Project via grant 2012AA011102. Special thanks to Shiqi Zhao, Zhenghua Li, Wei Song and the anonymous reviewers for insightful comments and suggestions. We also thank Xinwei Geng and Hongbo Cai for their help in the experiments.

References

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

Rahul Bhagat, Patrick Pantel, Eduard H Hovy, and Marina Rey. 2007. Ledir: An unsupervised algorithm for learning directionality of inference rules. In *EMNLP-CoNLL*, pages 161–170.

Wanxiang Che, Zhenghua Li, and Ting Liu. 2010. Ltp: A chinese language technology platform. In *Coling 2010: Demonstrations*, pages 13–16, Beijing, China, August.

Daoud Clarke. 2009. Context-theoretic semantics for natural language: an overview. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, pages 112–119. Association for Computational Linguistics.

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

Paramveer Dhillon, Dean P Foster, and Lyle H Ungar. 2011. Multi-view learning of word embeddings via cca. In *Advances in Neural Information Processing Systems*, pages 199–207.

Richard Evans. 2004. A framework for named entity recognition in the open domain. *Recent Advances in Natural Language Processing III: Selected Papers from RANLP 2003*, 260:267–274.

Ruiji Fu, Bing Qin, and Ting Liu. 2013. Exploiting multiple sources for open-domain hypernym discovery. In *EMNLP*, pages 1224–1234.

Maayan Geffet and Ido Dagan. 2005. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 107–114. Association for Computational Linguistics.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16(4):359–389.

Alessandro Lenci and Giulia Benotto. 2012. Identifying hypernyms in distributional semantic spaces. In *Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 75–79. Association for Computational Linguistics.

Paul McNamee, Rion Snow, Patrick Schone, and James Mayfield. 2008. Learning named entity hyponyms for question answering. In *Proceedings of the Third International Joint Conference on Natural Language Processing*, pages 799–804.

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

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

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Andriy Mnih and Geoffrey E Hinton. 2008. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088.

Rosa M Ortega-Mendoza, Luis Villaseñor-Pineda, and Manuel Montes-y Gómez. 2007. Using lexical patterns for extracting hyponyms from the web. In *MICAI 2007: Advances in Artificial Intelligence*, pages 904–911. Springer.

Alan Ritter, Stephen Soderland, and Oren Etzioni. 2009. What is this, anyway: Automatic hypernym discovery. In *Proceedings of the 2009 AAAI Spring Symposium on Learning by Reading and Learning to Read*, pages 88–93.

HP Ritzema. 1994. *Drainage principles and applications.*

Erik Tjong Kim Sang. 2007. Extracting hypernym pairs from the web. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 165–168. Association for Computational Linguistics.

Sidney Siegel and N John Castellan Jr. 1988. Nonparametric statistics for the behavioral sciences. McGraw-Hill, New York.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1297–1304. MIT Press, Cambridge, MA.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 801–808, Sydney, Australia, July. Association for Computational Linguistics.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Ng. 2011a. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011b. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2008. Yago: A large ontology from wikipedia and wordnet. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(3):203–217.

Idan Szpektor, Eyal Shnarch, and Ido Dagan. 2007. Instance-based evaluation of entailment rule acquisition. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 456–463, Prague, Czech Republic, June. Association for Computational Linguistics.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics.

Julie Weeds, David Weir, and Diana McCarthy. 2004. Characterising measures of lexical distributional similarity. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1015. Association for Computational Linguistics.

Probabilistic Soft Logic for Semantic Textual Similarity

Islam Beltagy[§] **Katrin Erk**[†] **Raymond Mooney**[§]
[§]Department of Computer Science
[†]Department of Linguistics
The University of Texas at Austin
Austin, Texas 78712
[§]{beltagy,mooney}@cs.utexas.edu
[†]katrin.erk@mail.utexas.edu

Abstract

Probabilistic Soft Logic (PSL) is a recently developed framework for probabilistic logic. We use PSL to combine logical and distributional representations of natural-language meaning, where distributional information is represented in the form of weighted inference rules. We apply this framework to the task of Semantic Textual Similarity (STS) (i.e. judging the semantic similarity of natural-language sentences), and show that PSL gives improved results compared to a previous approach based on Markov Logic Networks (MLNs) and a purely distributional approach.

1 Introduction

When will people say that two sentences are similar? This question is at the heart of the Semantic Textual Similarity task (STS)(Agirre et al., 2012). Certainly, if two sentences contain many of the same words, or many similar words, that is a good indication of sentence similarity. But that can be misleading. A better characterization would be to say that if two sentences use the same or similar words *in the same or similar relations*, then those two sentences will be judged similar.[1] Interestingly, this characterization echoes the principle of compositionality, which states that the meaning of a phrase is uniquely determined by the meaning of its parts and the rules that connect those parts.

Beltagy et al. (2013) proposed a hybrid approach to sentence similarity: They use a very deep representation of sentence meaning, expressed in first-order logic, to capture sentence structure, but combine it with distributional similarity ratings at the word and phrase level. Sentence similarity is then modelled as mutual entailment in a probabilistic logic. This approach is interesting in that it uses a very deep and precise representation of meaning, which can then be relaxed in a controlled fashion using distributional similarity. But the approach faces large hurdles in practice, stemming from efficiency issues with the Markov Logic Networks (MLN) (Richardson and Domingos, 2006) that they use for performing probabilistic logical inference.

In this paper, we use the same combined logic-based and distributional framework as Beltagy et al., (2013) but replace Markov Logic Networks with Probabilistic Soft Logic (PSL) (Kimmig et al., 2012; Bach et al., 2013). PSL is a probabilistic logic framework designed to have efficient inference. Inference in MLNs is theoretically intractable in the general case, and existing approximate inference algorithms are computationally expensive and sometimes inaccurate. Consequently, the MLN approach of Beltagy et al. (2013) was unable to scale to long sentences and was only tested on the relatively short sentences in the Microsoft video description corpus used for STS (Agirre et al., 2012). On the other hand, inference in PSL reduces to a linear programming problem, which is theoretically and practically much more efficient. Empirical results on a range of problems have confirmed that inference in PSL is much more efficient than in MLNs, and frequently more accurate (Kimmig et al., 2012; Bach et al., 2013).

We show how to use PSL for STS, and describe changes to the PSL framework that make it more effective for STS. For evaluation, we test on three STS datasets, and compare our PSL system with the MLN approach of Beltagy et al., (2013) and with distributional-only baselines. Experimental

[1] Mitchell and Lapata (2008) give an amusing example of two sentences that consist of all the same words, but are very different in their meaning: (a) It was not the sales manager who hit the bottle that day, but the office worker with the serious drinking problem. (b) That day the office manager, who was drinking, hit the problem sales worker with a bottle, but it was not serious.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1210–1219,
Baltimore, Maryland, USA, June 23-25 2014.

results demonstrate that, overall, PSL models human similarity judgements more accurately than these alternative approaches, and is significantly faster than MLNs.

The rest of the paper is organized as follows: section 2 presents relevant background material, section 3 explains how we adapted PSL for the STS task, section 4 presents the evaluation, and sections 5 and 6 discuss future work and conclusions, respectively.

2 Background

2.1 Logical Semantics

Logic-based representations of meaning have a long tradition (Montague, 1970; Kamp and Reyle, 1993). They handle many complex semantic phenomena such as relational propositions, logical operators, and quantifiers; however, their binary nature prevents them from capturing the "graded" aspects of meaning in language. Also, it is difficult to construct formal ontologies of properties and relations that have broad coverage, and semantically parsing sentences into logical expressions utilizing such an ontology is very difficult. Consequently, current semantic parsers are mostly restricted to quite limited domains, such as querying a specific database (Kwiatkowski et al., 2013; Berant et al., 2013). In contrast, our system is not limited to any formal ontology and can use a wide-coverage tool for semantic analysis, as discussed below.

2.2 Distributional Semantics

Distributional models (Turney and Pantel, 2010), on the other hand, use statistics on contextual data from large corpora to predict semantic similarity of words and phrases (Landauer and Dumais, 1997; Mitchell and Lapata, 2010). They are relatively easier to build than logical representations, automatically acquire knowledge from "big data," and capture the "graded" nature of linguistic meaning, but do not adequately capture logical structure (Grefenstette, 2013).

Distributional models are motivated by the observation that semantically similar words occur in similar contexts, so words can be represented as vectors in high dimensional spaces generated from the contexts in which they occur (Landauer and Dumais, 1997; Lund and Burgess, 1996). Such models have also been extended to compute vector representations for larger phrases, e.g. by adding the vectors for the individual words (Landauer and Dumais, 1997) or by a component-wise product of word vectors (Mitchell and Lapata, 2008; Mitchell and Lapata, 2010), or more complex methods that compute phrase vectors from word vectors and tensors (Baroni and Zamparelli, 2010; Grefenstette and Sadrzadeh, 2011). We use vector addition (Landauer and Dumais, 1997), and component-wise product (Mitchell and Lapata, 2008) as baselines for STS. Vector addition was previously found to be the best performing simple distributional method for STS (Beltagy et al., 2013).

2.3 Markov Logic Networks

Markov Logic Networks (MLN) (Richardson and Domingos, 2006) are a framework for probabilistic logic that employ weighted formulas in first-order logic to compactly encode complex undirected probabilistic graphical models (i.e., Markov networks). Weighting the rules is a way of softening them compared to hard logical constraints and thereby allowing situations in which not all clauses are satisfied. MLNs define a probability distribution over possible worlds, where a world's probability increases exponentially with the total weight of the logical clauses that it satisfies. A variety of inference methods for MLNs have been developed, however, developing a scalable, general-purpose, accurate inference method for complex MLNs is an open problem. Beltagy et al. (2013) use MLNs to represent the meaning of natural language sentences and judge textual entailment and semantic similarity, but they were unable to scale the approach beyond short sentences due to the complexity of MLN inference.

2.4 Probabilistic Soft Logic

Probabilistic Soft Logic (PSL) is a recently proposed alternative framework for probabilistic logic (Kimmig et al., 2012; Bach et al., 2013). It uses logical representations to compactly define large graphical models with continuous variables, and includes methods for performing efficient probabilistic inference for the resulting models. A key distinguishing feature of PSL is that ground atoms have soft, continuous truth values in the interval [0, 1] rather than binary truth values as used in MLNs and most other probabilistic logics. Given a set of weighted logical formulas, PSL builds a graphical model defining a probability distribution over the continuous space of values of the random variables in the model.

A PSL model is defined using a set of weighted if-then rules in first-order logic, as in the following example:

$$\forall x, y, z.\ friend(x, y) \wedge votesFor(y, z) \rightarrow votesFor(x, z) \mid 0.3 \quad (1)$$

$$\forall x, y, z.\ spouse(x, y) \wedge votesFor(y, z) \rightarrow votesFor(x, z) \mid 0.8 \quad (2)$$

In our notation, we use lower case letters like x, y, z to represent variables and upper case letters for constants. The first rule states that a person is likely to vote for the same person as his/her friend. The second rule encodes the same regularity for a person's spouse. The weights encode the knowledge that a spouse's influence is greater than a friend's in this regard.

In addition, PSL includes *similarity functions*. Similarity functions take two strings or two sets as input and return a truth value in the interval [0, 1] denoting the similarity of the inputs. For example, in our application, we generate inference rules that incorporate the similarity of two predicates. This can be represented in PSL as:

$$\forall x.\ similarity(\text{“}predicate1\text{”}, \text{“}predicate2\text{”}) \wedge predicate1(x) \rightarrow predicate2(x)$$

As mentioned above, each ground atom, a, has a soft truth value in the interval [0, 1], which is denoted by $I(a)$. To compute soft truth values for logical formulas, Lukasiewicz's relaxation of conjunctions($\wedge$), disjunctions($\vee$) and negations($\neg$) are used:

$$I(l_1 \wedge l_1) = max\{0, I(l_1) + I(l_2) - 1\}$$
$$I(l_1 \vee l_1) = min\{I(l_1) + I(l_2), 1\}$$
$$I(\neg l_1) = 1 - I(l_1)$$

Then, a given rule $r \equiv r_{body} \rightarrow r_{head}$, is said to be *satisfied* (i.e. $I(r) = 1$) iff $I(r_{body}) \leq I(r_{head})$. Otherwise, PSL defines a *distance to satisfaction* $d(r)$ which captures how far a rule r is from being satisfied: $d(r) = max\{0, I(r_{body}) - I(r_{head})\}$. For example, assume we have the set of evidence: $I(spouse(B, A)) = 1$, $I(votesFor(A, P)) = 0.9$, $I(votesFor(B, P)) = 0.3$, and that r is the resulting ground instance of rule (2). Then $I(spouse(B, A) \wedge votesFor(A, P)) = max\{0, 1 + 0.9 - 1\} = 0.9$, and $d(r) = max\{0, 0.9 - 0.3\} = 0.6$.

Using *distance to satisfaction*, PSL defines a probability distribution over all possible interpretations I of all ground atoms. The pdf is defined as follows:

$$p(I) = \frac{1}{Z} \exp\left[-\sum_{r \in R} \lambda_r (d(r))^p\right]; \quad (3)$$

$$Z = \int_I \exp\left[-\sum_{r \in R} \lambda_r (d(r))^p\right]$$

where Z is the normalization constant, λ_r is the weight of rule r, R is the set of all rules, and $p \in \{1, 2\}$ provides two different loss functions. For our application, we always use $p = 1$

PSL is primarily designed to support MPE inference (Most Probable Explanation). MPE inference is the task of finding the overall interpretation with the maximum probability given a set of evidence. Intuitively, the interpretation with the highest probability is the interpretation with the lowest distance to satisfaction. In other words, it is the interpretation that tries to satisfy all rules as much as possible. Formally, from equation 3, the most probable interpretation, is the one that minimizes $\sum_{r \in R} \lambda_r (d(r))^p$. In case of $p = 1$, and given that all $d(r)$ are linear equations, then minimizing the sum requires solving a linear program, which, compared to inference in other probabilistic logics such as MLNs, can be done relatively efficiently using well-established techniques. In case $p = 2$, MPE inference can be shown to be a second-order cone program (SOCP) (Kimmig et al., 2012).

2.5 Semantic Textual Similarity

Semantic Textual Similarity (STS) is the task of judging the similarity of a pair of sentences on a scale from 0 to 5, and was recently introduced as a SemEval task (Agirre et al., 2012). Gold standard scores are averaged over multiple human annotations and systems are evaluated using the Pearson correlation between a system's output and gold standard scores. The best performing system in 2012's competition was by Bär et al. (2012), a complex ensemble system that integrates many techniques including string similarity, n-gram overlap, WordNet similarity, vector space similarity and MT evaluation metrics. Two of the datasets we use for evaluation are from the 2012 competition. We did not utilize the new datasets added in the 2013 competition since they did not contain naturally-occurring, full sentences, which is the focus of our work.

2.6 Combining logical and distributional methods using probabilistic logic

There are a few recent attempts to combine logical and distributional representations in order to obtain the advantages of both. Lewis and Steedman (2013) use distributional information to determine word senses, but still produce a strictly logical semantic representation that does not address the "graded" nature of linguistic meaning that is important to measuring semantic similarity.

Garrette et al. (2011) introduced a framework for combining logic and distributional models using probabilistic logic. Distributional similarity between pairs of words is converted into weighted inference rules that are added to the logical representation, and Markov Logic Networks are used to perform probabilistic logical inference.

Beltagy et al. (2013) extended this framework by generating distributional inference rules from phrase similarity and tailoring the system to the STS task. STS is treated as computing the probability of two textual entailments $T \models H$ and $H \models T$, where T and H are the two sentences whose similarity is being judged. These two entailment probabilities are averaged to produce a measure of similarity. The MLN constructed to determine the probability of a given entailment includes the logical forms for both T and H as well as soft inference rules that are constructed from distributional information. Given a similarity score for all pairs of sentences in the dataset, a regressor is trained on the training set to map the system's output to the gold standard scores. The trained regressor is applied to the scores in the test set before calculating Pearson correlation. The regression algorithm used is Additive Regression (Friedman, 2002).

To determine an entailment probability, first, the two sentences are mapped to logical representations using Boxer (Bos, 2008), a tool for wide-coverage semantic analysis that maps a CCG (Combinatory Categorial Grammar) parse into a lexically-based logical form. Boxer uses C&C for CCG parsing (Clark and Curran, 2004).

Distributional semantic knowledge is then encoded as weighted inference rules in the MLN. A rule's weight (w) is a function of the cosine similarity (sim) between its antecedent and consequent. Rules are generated on the fly for each T and H. Let t and h be the lists of all words and phrases in T and H respectively. For all pairs (a, b), where $a \in t, b \in h$, it generates an inference rule: $a \rightarrow b \mid w$, where $w = f(sim(\overrightarrow{a}, \overrightarrow{b}))$. Both a and b can be words or phrases. Phrases are defined in terms of Boxer's output. A phrase is more than one unary atom sharing the same variable like "a little kid" which in logic is $little(K) \wedge kid(K)$. A phrase also can be two unary atoms connected by a relation like "a man is driving" which in logic is $man(M) \wedge agent(D, M) \wedge drive(D)$. The similarity function sim takes two vectors as input. Phrasal vectors are constructed using Vector Addition (Landauer and Dumais, 1997). The set of generated inference rules can be regarded as the knowledge base KB.

Beltagy et al. (2013) found that the logical conjunction in H is very restrictive for the STS task, so they relaxed the conjunction by using an average evidence combiner (Natarajan et al., 2010). The average combiner results in computationally complex inference and only works for short sentences. In case inference breaks or times-out, they back off to a simpler combiner that leads to much faster inference but loses most of the structure of the sentence and is therefore less accurate.

Given T, KB and H from the previous steps, MLN inference is then used to compute $p(H|T, KB)$, which is then used as a measure of the degree to which T entails H.

3 PSL for STS

For several reasons, we believe PSL is a more appropriate probabilistic logic for STS than MLNs. First, it is explicitly designed to support efficient inference, therefore it scales better to longer sentences with more complex logical forms. Second, it was also specifically designed for computing *similarity* between complex structured objects rather than determining probabilistic logical entailment. In fact, the initial version of PSL (Broecheler et al., 2010) was called *Probabilistic Similarity Logic*, based on its use of *similarity functions*. This initial version was shown to be very effective for measuring the similarity of noisy database records and performing *record linkage* (i.e. identifying database entries referring to the same entity, such as bibliographic citations referring to the same paper). Therefore, we have developed an approach that follows that of Beltagy et al. (2013), but replaces Markov Logic with PSL.

This section explains how we formulate the STS

task as a PSL program. PSL does not work very well "out of the box" for STS, mainly because Lukasiewicz's equation for the conjunction is very restrictive. Therefore, we use a different interpretation for conjunction that uses averaging, which requires corresponding changes to the optimization problem and the grounding technique.

3.1 Representation

Given the logical forms for a pair of sentences, a text T and a hypothesis H, and given a set of weighted rules derived from the distributional semantics (as explained in section 2.6) composing the knowledge base KB, we build a PSL model that supports determining the truth value of H in the most probable interpretation (i.e. MPE) given T and KB.

Consider the pair of sentences is "A man is driving", and "A guy is walking". Parsing into logical form gives:

$T: \exists x, y.\ man(x) \wedge agent(y, x) \wedge drive(y)$

$H: \exists x, y.\ guy(x) \wedge agent(y, x) \wedge walk(y)$

The PSL program is constructed as follows:

T: The text is represented in the evidence set. For the example, after Skolemizing the existential quantifiers, this contains the ground atoms: $\{man(A), agent(B, A), drive(B)\}$

KB: The knowledge base is a set of lexical and phrasal rules generated from distributional semantics, along with a similarity score for each rule (section 2.6). For the example, we generate the rules: $\forall x.\ man(x) \wedge vs\_sim(\text{"}man\text{"}, \text{"}guy\text{"}) \rightarrow guy(x)$, $\forall x. drive(x) \wedge vs\_sim(\text{"}drive\text{"}, \text{"}walk\text{"}) \rightarrow walk(x)$

where vs_sim is a similarity function that calculates the distributional similarity score between the two lexical predicates. All rules are assigned the same weight because all rules are equally important.

H: The hypothesis is represented as $H \rightarrow result()$, and then PSL is queried for the truth value of the atom $result()$. For our example, the rule is: $\forall x, y.\ guy(x) \wedge agent(y, x) \wedge walk(y) \rightarrow result()$.

Priors: A low prior is given to all predicates. This encourages the truth values of ground atoms to be zero, unless there is evidence to the contrary.

For each STS pair of sentences S_1, S_2, we run PSL twice, once where $T = S_1$, $H = S_2$ and another where $T = S_2$, $H = S_1$, and output the two scores. To produce a final similarity score, we train a regressor to learn the mapping between the two PSL scores and the overall similarity score. As in Beltagy et al., (2013) we use Additive Regression (Friedman, 2002).

3.2 Changing Conjunction

As mentioned above, Lukasiewicz's formula for conjunction is very restrictive and does not work well for STS. For example, for T: "A man is driving" and H: "A man is driving a car", if we use the standard PSL formula for conjunction, the output value is zero because there is no evidence for a car and $max(0, X + 0 - 1) = 0$ for any truth value $0 \leq X \leq 1$. However, humans find these sentences to be quite similar.

Therefore, we introduce a new averaging interpretation of conjunction that we use for the hypothesis H. The truth value for a conjunction is defined as $I(p_1 \wedge .... \wedge p_n) = \frac{1}{n} \sum_{i=1}^{n} I(p_i)$. This averaging function is linear, and the result is a valid truth value in the interval [0, 1], therefore this change is easily incorporated into PSL without changing the complexity of inference which remains a linear-programming problem.

It would perhaps be even better to use a weighted average, where weights for different components are learned from a supervised training set. This is an important direction for future work.

3.3 Grounding Process

Grounding is the process of instantiating the variables in the quantified rules with concrete constants in order to construct the nodes and links in the final graphical model. In principle, grounding requires instantiating each rule in all possible ways, substituting every possible constant for each variable in the rule. However, this is a combinatorial process that can easily result in an explosion in the size of the final network. Therefore, PSL employs a "lazy" approach to grounding that avoids the construction of irrelevant groundings. If there is no evidence for one of the antecedents in a particular grounding of a rule, then the normal PSL formula for conjunction guarantees that the rule is

Algorithm 1 Heuristic Grounding

Input: $r_{body} = a_1 \wedge .... \wedge a_n$: antecedent of a rule with average interpretation of conjunction
Input: V: set of variables used in r_{body}
Input: $Ant(v_i)$: subset of antecedents a_j containing variable v_i
Input: $Const(v_i)$: list of possible constants of variable v_i
Input: $Gnd(a_i)$: set of ground atoms of a_i.
Input: $GndConst(a, g, v)$: takes an atom a, grounding g for a, and variable v, and returns the constant that substitutes v in g
Input: gnd_limit: limit on the number of groundings
1: **for all** $v_i \in V$ **do**
2: **for all** $C \in Const(v_i)$ **do**
3: $score(C) = \sum_{a \in Ant(v_i)}(max\ I(g))$ for $g \in Gnd(a) \wedge GndConst(a, g, v_i) = C$
4: **end for**
5: sort $Const(v_i)$ on scores, descending
6: **end for**
7: **return** For all $v_i \in V$, take the Cartesian-product of the sorted $Const(v_i)$ and return the top gnd_limit results

trivially satisfied ($I(r) = 1$) since the truth value of the antecedent is zero. Therefore, its distance to satisfaction is also zero, and it can be omitted from the ground network without impacting the result of MPE inference.

However, this technique does not work once we switch to using averaging to interpret conjunctions. For example, given the rule $\forall x.\ p(x) \wedge q(x) \rightarrow t()$ and only one piece of evidence $p(C)$ there are no relevant groundings because there is no evidence for $q(C)$, and therefore, for normal PSL, $I(p(C) \wedge q(C)) = 0$ which does not affect $I(t())$. However, when using averaging with the same evidence, we need to generate the grounding $p(C) \wedge q(C)$ because $I(p(C) \wedge q(C)) = 0.5$ which *does* affect $I(t())$.

One way to solve this problem is to eliminate lazy grounding and generate all possible groundings. However, this produces an intractably large network. Therefore, we developed a heuristic approximate grounding technique that generates a subset of the most impactful groundings.

Pseudocode for this heuristic approach is shown in algorithm 1. Its goal is to find constants that participate in ground propositions with high truth value and preferentially use them to construct a limited number of groundings of each rule.

The algorithm takes the antecedents of a rule employing averaging conjunction as input. It also takes the *grounding limit* which is a threshold on the number of groundings to be returned. The algorithm uses several subroutines, they are:

- $Ant(v_i)$: given a variable v_i, it returns the set of rule antecedent atoms containing v_i. E.g, for the rule: $a(x) \wedge b(y) \wedge c(x)$, $Ant(x)$ returns the set of atoms $\{a(x), c(x)\}$.
- $Const(v_i)$: given a variable v_i, it returns the list of possible constants that can be used to instantiate the variable v_i.
- $Gnd(a_i)$: given an atom a_i, it returns the set of all possible ground atoms generated for a_i.
- $GndConst(a, g, v)$: given an atom a and grounding g for a, and a variable v, it finds the constant that substitutes for v in g. E.g, assume there is an atom $a = a_i(v_1, v_2)$, and the ground atom $g = a_i(A, B)$ is one of its groundings. $GndConst(a, g, v_2)$ would return the constant B since it is the substitution for the variable v_2 in g.

Lines 1-6 loop over all variables in the rule. For each variable, lines 2-5 construct a list of constants for that variable and sort it based on a heuristic score. In line 3, each constant is assigned a score that indicates the importance of this constant in terms of its impact on the truth value of the overall grounding. A constant's score is the sum, over all antedents that contain the variable in question, of the maximum truth value of any grounding of that antecedent that contains that constant.

Pushing constants with high scores to the top of each variable's list will tend to make the overall truth value of the top groundings high. Line 7 computes a subset of the Cartesian product of the sorted lists of constants, selecting constants in ranked order and limiting the number of results to the grounding limit.

One point that needs to be clarified about this approach is how it relies on the truth values of ground atoms when the goal of inference is to actually find these values. PSL's inference is actually an iterative process where in each iteration a grounding phase is followed by an optimization phase (solving the linear program). This loop repeats until convergence, i.e. until the truth

values stop changing. The truth values used in each grounding phase come from the previous optimization phase. The first grounding phase assumes only the propositions in the evidence provided have non-zero truth values.

4 Evaluation

This section evaluates the performance of PSL on the STS task.

4.1 Datasets

We evaluate our system on three STS datasets.

- **msr-vid**: Microsoft Video Paraphrase Corpus from STS 2012. The dataset consists of 1,500 pairs of short video descriptions collected using crowdsourcing (Chen and Dolan, 2011) and subsequently annotated for the STS task (Agirre et al., 2012). Half of the dataset is for training, and the second half is for testing.

- **msr-par**: Microsoft Paraphrase Corpus from STS 2012 task. The dataset is 5,801 pairs of sentences collected from news sources (Dolan et al., 2004). Then, for STS 2012, 1,500 pairs were selected and annotated with similarity scores. Half of the dataset is for training, and the second half is for testing.

- **SICK**: Sentences Involving Compositional Knowledge is a dataset collected for SemEval 2014. Only the training set is available at this point, which consists of 5,000 pairs of sentences. Pairs are annotated for RTE and STS, but we only use the STS data. Training and testing was done using 10-fold cross validation.

4.2 Systems Compared

We compare our PSL system with several others. In all cases, we use the distributional word vectors employed by Beltagy et al. (2013) based on context windows from Gigaword.

- **vec-add**: Vector Addition (Landauer and Dumais, 1997). We compute a vector representation for each sentence by adding the distributional vectors of all of its words and measure similarity using cosine. This is a simple yet powerful baseline that uses only distributional information.

- **vec-mul**: Component-wise Vector Multiplication (Mitchell and Lapata, 2008). The same as **vec-add** except uses component-wise multiplication to combine word vectors.

- **MLN**: The system of Beltagy et al. (2013), which uses Markov logic instead of PSL for probabilistic inference. MLN inference is very slow in some cases, so we use a 10 minute timeout. When MLN times out, it backs off to a simpler sentence representation as explained in section 2.6.

- **PSL**: Our proposed PSL system for combining logical and distributional information.

- **PSL-no-DIR**: Our PSL system without distributional inference rules(empty knowledge base). This system uses PSL to compute similarity of logical forms but does not use distributional information on lexical or phrasal similarity. It tests the impact of the probabilistic logic only

- **PSL+vec-add**: **PSL** ensembled with **vec-add**. Ensembling the MLN approach with a purely distributional approach was found to improve results (Beltagy et al., 2013), so we also tried this with PSL. The methods are ensembled by using both entailment scores of both systems as input features to the regression step that learns to map entailment scores to STS similarity ratings. This way, the training data is used to learn how to weight the contribution of the different components.

- **PSL+MLN**: **PSL** ensembled with **MLN** in the same manner.

4.3 Experiments

Systems are evaluated on two metrics, Pearson correlation and average CPU time per pair of sentences.

- Pearson correlation: The Pearson correlation between the system's similarity scores and the human gold-standards.

- CPU time: This metric only applies to MLN and PSL. The CPU time taken by the inference step is recorded and averaged over all pairs in each of the test datasets. In many cases, MLN inference is very slow, so we timeout after 10 minutes and report the number of timed-out pairs on each dataset.

	msr-vid	msr-par	SICK
vec-add	0.78	0.24	0.65
vec-mul	0.76	0.12	0.62
MLN	0.63	0.16	0.47
PSL-no-DIR	0.74	0.46	0.68
PSL	0.79	**0.53**	0.70
PSL+vec-add	**0.83**	0.49	**0.71**
PSL+MLN	0.79	0.51	0.70
Best Score (Bär et al., 2012)	0.87	0.68	n/a

Table 1: STS Pearson Correlations

	PSL	MLN	
	time	time	timeouts/total
msr-vid	8s	1m 31s	132/1500
msr-par	30s	11m 49s	1457/1500
SICK	10s	4m 24s	1791/5000

Table 2: Average CPU time per STS pair, and number of timed-out pairs in MLN with a 10 minute time limit. PSL's grounding limit is set to 10,000 groundings.

We also evaluated the effect of changing the grounding limit on both Pearson correlation and CPU time for the **msr-par** dataset. Most of the sentences in **msr-par** are long, which results is large number of groundings, and limiting the number of groundings has a visible effect on the overall performance. In the other two datasets, the sentences are fairly short, and the full number of groundings is not large; therefore, changing the grounding limit does not significantly affect the results.

4.4 Results and Discussion

Table 1 shows the results for Pearson correlation. **PSL** out-performs the purely distributional baselines (**vec-add** and **vec-mul**) because **PSL** is able to combine the information available to **vec-add** and **vec-mul** in a better way that takes sentence structure into account. **PSL** also outperforms the unaided probabilistic-logic baseline that does not use distributional information (**PSL-no-DIR**). **PSL-no-DIR** works fairly well because there is significant overlap in the exact words and structure of the paired sentences in the test data, and PSL combines the evidence from these similarities effectively. In addition, **PSL** always does significantly better than **MLN**, because of the large

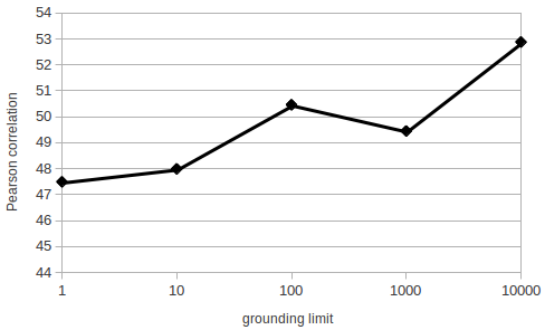


Figure 1: Effect of **PSL**'s grounding limit on the correlation score for the **msr-par** dataset

number of timeouts, and because the conjunction-averaging in PSL is combining evidence better than MLN's average-combiner, whose performance is sensitive to various parameters. These results further support the claim that using probabilistic logic to integrate logical and distributional information is a promising approach to natural-language semantics. More specifically, they strongly indicate that PSL is a more effective probabilistic logic for judging semantic similarity than MLNs.

Like for MLNs (Beltagy et al., 2013), ensembling PSL with vector addition improved the scores a bit, except for **msr-par** where **vec-add**'s performance is particularly low. However, this ensemble still does not beat the state of the art (Bär et al., 2012) which is a large ensemble of many different systems. It would be informative to add our system to their ensemble to see if it could improve it even further.

Table 2 shows the CPU time for **PSL** and **MLN**. The results clearly demonstrate that **PSL** is an order of magnitude faster than **MLN**.

Figures 1 and 2 show the effect of changing the grounding limit on Pearson correlation and CPU time. As expected, as the grounding limit is increased, accuracy improves but CPU time also increases. However, note that the difference in scores between the smallest and largest grounding limit tested is not large, suggesting that the heuristic approach to limiting grounding is quite effective.

5 Future Work

As mentioned in Section 3.2, it would be good to use a *weighted* average to compute the truth

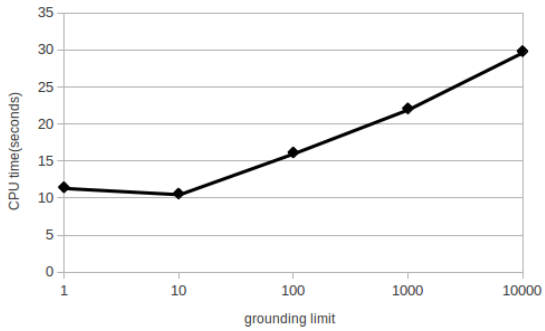


Figure 2: Effect of **PSL**'s grounding limit on CPU time for the **msr-par** dataset

values for conjunctions, weighting some predicates more than others rather than treating them all equally. Appropriate weights for different components could be learned from training data. For example, such an approach could learn that the type of an object determined by a noun should be weighted more than a property specified by an adjective. As a result, "black dog" would be appropriately judged more similar to "white dog" than to "black cat."

One of the advantages of using a probabilistic logic is that additional sources of knowledge can easily be incorporated by adding additional soft inference rules. To complement the soft inference rules capturing distributional lexical and phrasal similarities, PSL rules could be added that encode explicit paraphrase rules, such as those mined from monolingual text (Berant et al., 2011) or multi-lingual parallel text (Ganitkevitch et al., 2013).

This paper has focused on STS; however, as shown by Beltagy et al. (2013), probabilistic logic is also an effective approach to *recognizing textual entailment* (RTE). By using the appropriate functions to combine truth values for various logical connectives, PSL could also be adapted for RTE. Although we have shown that PSL outperforms MLNs on STS, we hypothesize that MLNs may still be a better approach for RTE. However, it would be good to experimentally confirm this intuition. In any case, the high computational complexity of MLN inference could mean that PSL is still a more practical choice for RTE.

6 Conclusion

This paper has presented an approach that uses Probabilistic Soft Logic (PSL) to determine Semantic Textual Similarity (STS). The approach uses PSL to effectively combine logical semantic representations of sentences with soft inference rules for lexical and phrasal similarities computed from distributional information. The approach builds upon a previous method that uses Markov Logic (MLNs) for STS, but replaces the probabilistic logic with PSL in order to improve the efficiency and accuracy of probabilistic inference. The PSL approach was experimentally evaluated on three STS datasets and was shown to outperform purely distributional baselines as well as the MLN approach. The PSL approach was also shown to be much more scalable and efficient than using MLNs

Acknowledgments

This research was supported by the DARPA DEFT program under AFRL grant FA8750-13-2-0026. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the view of DARPA, DoD or the US government. Some experiments were run on the Mastodon Cluster supported by NSF Grant EIA-0303609.

References

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of Semantic Evaluation (SemEval-12)*.

Stephen H. Bach, Bert Huang, Ben London, and Lise Getoor. 2013. Hinge-loss Markov random fields: Convex inference for structured prediction. In *Proceedings of Uncertainty in Artificial Intelligence (UAI-13)*.

Daniel Bär, Chris Biemann, Iryna Gurevych, and Torsten Zesch. 2012. UKP: Computing semantic textual similarity by combining multiple content similarity measures. In *Proceedings of Semantic Evaluation (SemEval-12)*.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-10)*.

Islam Beltagy, Cuong Chau, Gemma Boleda, Dan Garrette, Katrin Erk, and Raymond Mooney. 2013.

Montague meets Markov: Deep semantics with probabilistic logical form. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics (*SEM-13)*.

Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2011. Global learning of typed entailment rules. In *Proceedings of Association for Computational Linguistics (ACL-11)*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-13)*.

Johan Bos. 2008. Wide-coverage semantic analysis with Boxer. In *Proceedings of Semantics in Text Processing (STEP-08)*.

Matthias Broecheler, Lilyana Mihalkova, and Lise Getoor. 2010. Probabilistic Similarity Logic. In *Proceedings of Uncertainty in Artificial Intelligence (UAI-20)*.

David L. Chen and William B. Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of Association for Computational Linguistics (ACL-11)*.

Stephen Clark and James R. Curran. 2004. Parsing the WSJ using CCG and log-linear models. In *Proceedings of Association for Computational Linguistics (ACL-04)*.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the International Conference on Computational Linguistics (COLING-04)*.

Jerome H Friedman. 2002. Stochastic gradient boosting. *Journal of Computational Statistics & Data Analysis (CSDA-02)*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-13)*.

Dan Garrette, Katrin Erk, and Raymond Mooney. 2011. Integrating logical representations with probabilistic information using Markov logic. In *Proceedings of International Conference on Computational Semantics (IWCS-11)*.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-11)*.

Edward Grefenstette. 2013. Towards a formal distributional semantics: Simulating logical calculi with tensors. In *Proceedings of Second Joint Conference on Lexical and Computational Semantics (*SEM 2013)*.

Hans Kamp and Uwe Reyle. 1993. *From Discourse to Logic*. Kluwer.

Angelika Kimmig, Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. 2012. A short introduction to Probabilistic Soft Logic. In *Proceedings of NIPS Workshop on Probabilistic Programming: Foundations and Applications (NIPS Workshop-12)*.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-13)*.

T. K. Landauer and S. T. Dumais. 1997. A solution to Plato's problem: The Latent Semantic Analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review*.

Mike Lewis and Mark Steedman. 2013. Combined distributional and logical semantics. *Transactions of the Association for Computational Linguistics (TACL-13)*.

Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, and Computers*.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of Association for Computational Linguistics (ACL-08)*.

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

Richard Montague. 1970. Universal grammar. *Theoria*, 36:373–398.

Sriraam Natarajan, Tushar Khot, Daniel Lowd, Prasad Tadepalli, Kristian Kersting, and Jude Shavlik. 2010. Exploiting causal independence in Markov logic networks: Combining undirected and directed models. In *Proceedings of European Conference in Machine Learning (ECML-10)*.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine Learning*, 62:107–136.

Peter Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research (JAIR-10)*.

Abstractive Summarization of Spoken and Written Conversations Based on Phrasal Queries

Yashar Mehdad **Giuseppe Carenini** **Raymond T. Ng**
Department of Computer Science, University of British Columbia
Vancouver, BC, V6T 1Z4, Canada
{mehdad, carenini, rng}@cs.ubc.ca

Abstract

We propose a novel abstractive query-based summarization system for conversations, where queries are defined as phrases reflecting a user information needs. We rank and extract the utterances in a conversation based on the overall content and the phrasal query information. We cluster the selected sentences based on their lexical similarity and aggregate the sentences in each cluster by means of a word graph model. We propose a ranking strategy to select the best path in the constructed graph as a query-based abstract sentence for each cluster. A resulting summary consists of abstractive sentences representing the phrasal query information and the overall content of the conversation. Automatic and manual evaluation results over meeting, chat and email conversations show that our approach significantly outperforms baselines and previous extractive models.

1 Introduction

Our lives are increasingly reliant on multimodal conversations with others. We email for business and personal purposes, attend meetings in person, chat online, and participate in blog or forum discussions. While this growing amount of personal and public conversations represent a valuable source of information, going through such overwhelming amount of data, to satisfy a particular information need, often leads to an information overload problem (Jones et al., 2004). Automatic summarization has been proposed in the past as a way to address this problem (e.g., (Sakai and Sparck-Jones, 2001)). However, often a good summary cannot be generic and should be a brief and well-organized paragraph that answer a user's information need.

The Document Understanding Conference (DUC)[1] has launched query-focused multidocument summarization as its main task since 2004, by focusing on complex queries with very specific answers. For example, "*How were the bombings of the US embassies in Kenya and Tanzania conducted? How and where were the attacks planned?*". Such complex queries are appropriate for a user who has specific information needs and can formulate the questions precisely. However, especially when dealing with conversational data that tend to be less structured and less topically focused, a user is often initially only exploring the source documents, with less specific information needs. Moreover, following the common practice in search engines, users are trained to form simpler and shorter queries (Meng and Yu, 2010). For example, when a user is interested in certain characteristics of an entity in online reviews (e.g., "*location*" or "*screen*") or a specific entity in a blog discussion (e.g., "*new model of iphone*"), she would not initially compose a complex query.

To address these issues, in this work, we tackle the task of conversation summarization based on *phrasal* queries. We define a phrasal query as a concatenation of two or more keywords, which is a more realistic representation of a user's information needs. For conversational data, this definition is more similar to the concept of search queries in information retrieval systems as well as to the concept of topic labels in the task of topic modeling. Example 1 shows two queries and their associated human written summaries based on a single chat log. We can observe that the two summaries, although generated from the same chat log, are totally distinct. This further demonstrates the importance of phrasal query-based summarization systems for long conversations.

To date, most systems in the area of summa-

[1] http://www-nlpir.nist.gov/projects/duc/index.html

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1220–1230,
Baltimore, Maryland, USA, June 23-25 2014.

Query-1: Test/Sample database for GNUe
Abstract-1: James Thompson asked Reinhard: I was going to work on the sample tonight. You mentioned wanting a fishhook and all data types. Any other things you want to see in there? Reinhard said that master/detail would be good, as there have been bugs only appearing in 3-level case. James said he already included that and I know I need to add a boolean. Did you want date as well as date-time? Reinhard said yes - we also have time values (time without date). They are especially interesting. James had not ever had use for something like that so I'm not sure where I would graft that in.
Query-2: Passing parameters to Forms
Abstract-2: James Thompson (jamest) asked how did parameter support in forms change recently? He reported the trigger namespace function referencesGFForm.parameters - which no longer exists. Reinhard said every GFForm should have a parameters. James said he was using parameters in on-startup. Reinhard said that's probably the only place where they don't work. James said that I'm thinking about moving that to on-activation instead of on-startup anyway as it should still work for a main form - but i still wonder if the on-startup parameter issue should be considered a bug - as it shouldn't choke. Reinhard was sure it should be considered a bug but I have no idea how to fix it. We haven't found a way to deal with parameters that works for every case. I don't know if there is any chance to pass the parameters to the form before it is activated. James asked how are parameters handled now? Reinhard replied that they are passed to activateForm so they are available from activation for the –main– form, the command line parameters are passed and for dialogs, the parameters are passed that were given in runDialog.

Example 1: Sample queries and associated human-written query-based summaries for a chat log.

rization focus on news or other well-written documents, while research on summarizing multiparty written conversations (e.g., chats, emails) has been limited. This is because traditional NLP approaches developed for formal texts often are not satisfactory when dealing with multiparty written conversations, which are typically in a casual style and do not display a clear syntactic structure with proper grammar and spelling. Even though some works try to address the problem of summarizing multiparty written conversions (e.g., (Mehdad et al., 2013b; Wang and Cardie, 2013; Murray et al., 2010; Zhou and Hovy, 2005; Gillick et al., 2009)), they do so in a generic way (not query-based) and focus on only one conversational domain (e.g., meetings). Moreover, most of the proposed systems for conversation summarization are extractive.

To address such limitations, we propose a fully automatic *unsupervised* abstract generation framework based on phrasal queries for multimodal conversation summarization. Our key contributions in this work are as follows:

1) To the best of our knowledge, our framework is the first abstractive system that generates summaries based on users phrasal queries, instead of well-formed questions. As a by-product of our approach, we also propose an extractive summarization model based on phrasal queries to select the summary-worthy sentences in the conversation based on query terms and signature terms (Lin and Hovy, 2000).

2) We propose a novel ranking strategy to select the best path in the constructed word graph by taking the query content, overall information content and grammaticality (i.e., fluency) of the sentence into consideration.

3) Although most of the current summarization approaches use supervised algorithms as a part of their system (e.g., (Wang et al., 2013)), our method can be totally unsupervised and does not depend on human annotation.

4) Although different conversational modalities (e.g., email vs. chat vs. meeting) underline domain-specific characteristics, in this work, we take advantage of their underlying similarities to generalize away from specific modalities and determine effective method for query-based summarization of multimodal conversations.

We evaluate our system over GNUe Traffic archive[2] Internet Relay Chat (IRC) logs, AMI meetings corpus (Carletta et al., 2005) and BC3 emails dataset (Ulrich et al., 2008). Automatic evaluation on the chat dataset and manual evaluation over the meetings and emails show that our system uniformly and statistically significantly outperforms baseline systems, as well as a state-of-the-art query-based extractive summarization system.

2 Phrasal Query Abstraction Framework

Our phrasal query abstraction framework generates a grammatical abstract from a conversation following three steps, as shown in Figure 1.

2.1 Utterance Extraction

Abstractive summary sentences can be created by aggregating and merging multiple sentences into an abstract sentence. In order to generate such a sentence, we need to identify which sentences from the original document should be extracted and combined to generate abstract sentences. In other words, we want to identify the summary-worthy sentences in the text that can be combined into an abstract sentence. This task can be considered as content selection. Moreover, this step, stand alone, corresponds to an extractive summarization system.

[2] http://kt.earth.li/GNUe/index.html

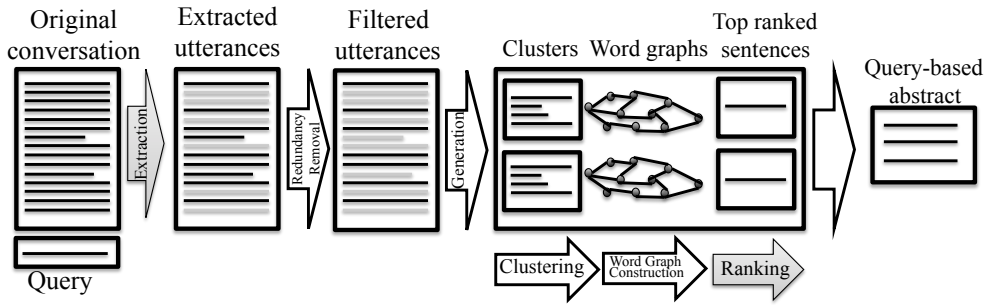


Figure 1: Phrasal query abstraction framework. The steps (arrows) influenced by the query are highlighted.

Signature terms: navigator, functionality, reports, UI, schema, gnu

Chat log:
- but watching them build a UI in the flash demo's is pretty damn impressive... and have started moving my sales app to all UI being built via ...
- i'll be expanding the technotes in navigator for a while ...
- ... in terms of functionality of the underlying databases ...
- you mean if I start GNU again I have to read bug reports too?
- no, just in case you want to enter bug report
- ...I expand the schema before populating with test data ...
- i'm willing to scrap it if there is a better schema hidden in gnue somewhere :)

Example 2: Sample signature terms for a part of a chat log.

In order to select and extract the informative summary-worthy utterances, based on the phrasal query and the original text, we consider two criteria: *i)* utterances should carry the essence of the original text; and *ii)* utterances should be relevant to the query. To fulfill such requirements we define the concepts of signature terms and query terms.

2.1.1 Signature Terms

Signature terms are generally indicative of the content of a document or collection of documents. To identify such terms, we can use frequency, word probability, standard statistic tests, information-theoretic measures or log-likelihood ratio. In this work, we use log-likelihood ratio to extract the signature terms from chat logs, since log-likelihood ratio leads to better results (Gupta et al., 2007). We use a method described in (Lin and Hovy, 2000) in order to identify such terms and their associated weight. Example 2 demonstrates a chat log and associated signature terms.

2.1.2 Query Terms

Query terms are indicative of the content in a phrasal query. In order to identify such terms, we first extract all content terms from the query. Then, following previous studies (e.g., (Gonzalo et al., 1998)), we use the synsets relations in WordNet for query expansion. We extract all concepts that are synonyms to the query terms and add them to the original set of query terms. Note that we limit our synsets to the nouns since verb synonyms do not prove to be effective in query expansion (Hunemark, 2010). While signature terms are weighted, we assume that all query terms are equally important and they all have wight equal to 1.

2.1.3 Utterance Scoring

To estimate the utterance score, we view both the query terms and the signature terms as the terms that should appear in a human query-based summary. To achieve this, the most relevant (summary-worthy) utterances that we select are the ones that maximize the coverage of such terms. Given the query terms and signature terms, we can estimate the utterance score as follows:

$$Score_Q = \frac{1}{n}\sum_{i=1}^{n} t(q)_i \quad (1)$$

$$Score_S = \frac{1}{n}\sum_{i=1}^{n} t(s)_i \times w(s)_i \quad (2)$$

$$Score = \alpha \cdot Score_Q + \beta \cdot Score_S \quad (3)$$

where n is number of content words in the utterance, $t(q)_i = 1$ if the term t_i is a query term and 0 otherwise, and $t(s)_i = 1$ if t_i is a signature term and 0 otherwise, and $w(s)_i$ is the normalized associated weight for signature terms. The parameters α and β are tuned on a development set and sum up to 1.

After all the utterances are scored, the top scored utterances are selected to be sent to the next step. We estimate the percentage of the retrieved utterances based on the development set.

2.2 Redundancy Removal

Utterances selected in previous step often include redundant information, which is semantically equivalent but may vary in lexical choices. By identifying the semantic relations between the sentences, we can discover what information in one sentence is semantically equivalent, novel, or more/less informative with respect to the content of the other sentences. Similar to earlier work (Berant et al., 2011; Adler et al., 2012), we set this problem as a variant of the Textual Entailment (TE) recognition task (Dagan and Glickman, 2004). Using entailment in this phase is motivated by taking advantage of semantic relations instead of pure statistical methods (e.g., Maximal Marginal Relevance) and shown to be more effective (Mehdad et al., 2013a). We follow the same practice as (Mehdad et al., 2013a) to build an entailment graph for all selected sentences to identify relevant sentences and eliminate the redundant (in terms of meaning) and less informative ones.

2.3 Abstract Generation

In this phase, our goal is to generate understandable informative abstract sentences that capture the content of the source sentences and represents the information needs defined by queries. There are several ways of generating abstract sentences (e.g. (Barzilay and McKeown, 2005; Liu and Liu, 2009; Ganesan et al., 2010; Murray et al., 2010)); however, most of them rely heavily on the sentence structure. We believe that such approaches are suboptimal, especially in dealing with conversational data, because multiparty written conversations are often poorly structured. Instead, we apply an approach that does not rely on syntax, nor on a standard NLG architecture. Moreover, since dealing with user queries efficiency is an important aspect, we aim for an approach that is also motivated by the speed with which the abstracts are obtained. We perform the task of abstract generation in three steps, as follows:

2.3.1 Clustering

In order to generate an abstract summary, we need to identify which sentences from the previous step (i.e., redundancy removal) can be clustered and combined in generated abstract sentences. This task can be viewed as sentence clustering, where each sentence cluster can provide the content for an abstract sentence.

We use the K-mean clustering algorithm by cosine similarity as a distance function between sentence vectors composed of *tf.idf* scores. Also notice that the lexical similarity between sentences in one cluster facilitates both the construction of the word graph and finding the best path in the word graph, as described next.

2.3.2 Word Graph

In order to construct a word graph, we adopt the method recently proposed by (Mehdad et al., 2013a; Filippova, 2010) with some optimizations. Below, we show how the word graph is applied to generate the abstract sentences.

Let $G = (W, L)$ be a directed graph with the set of nodes W representing words and a set of directed edges L representing the links between words. Given a cluster of related sentences $S = \{s_1, s_2, ..., s_n\}$, a word graph is constructed by iteratively adding sentences to it. In the first step, the graph represents one sentence plus the start and end symbols. A node is added to the graph for each word in the sentence, and words adjacent are linked with directed edges. When adding a new sentence, a word from the sentence is merged in an existing node in the graph providing that they have the same POS tag and they satisfy one of the following conditions:

i) They have the same word form;

ii) They are connected in WordNet by the synonymy relation. In this case the lexical choice for the node is selected based on the *tf.idf* score of each node;

iii) They are from a hypernym/hyponym pair or share a common direct hypernym. In this case, both words are replaced by the hypernym;

iv) They are in an entailment relation. In this case, the entailing word is replaced by the entailed one.

The motivation behind merging non-identical words is to enrich the common terms between the phrases to increase the chance that they could merge into a single phrase. This also helps to move beyond the limitation of original lexical choices. In case the merging is not possible a new node is created in the graph. When a node can be merged with multiple nodes (i.e., merging is ambiguous), either the preceding and following words in the sentence and the neighboring nodes in the graph or the frequency is used to select the candidate node.

We connect adjacent words with directed edges.

For the new nodes or unconnected nodes, we draw an edge with a weight of 1. In contrast, when two already connected nodes are added (merged), the weight of their connection is increased by 1.

2.3.3 Path Ranking

A word graph, as described above, may contain many sequences connecting start and end. However, it is likely that most of the paths are not readable. We are aiming at generating an informative abstractive sentence for each cluster based on a user query. Moreover, the abstract sentence should be grammatically correct.

In order to satisfy both requirements, we have devised the following ranking strategy. First, we prune the paths in which a verb does not exist, to filter ungrammatical sentences. Then we rank other paths as follows:

Query focus: to identify the summary sentence with the highest coverage of query content, we propose a score that counts the number of query terms that appear in the path. In order to reward the ranking score to cover more salient terms in the query content, we also consider the *tf.idf* score of query terms in the coverage formulation.

$$Q(P) = \frac{\sum_{q_i \in P} tfidf(q_i)}{\sum_{q_i \in G} tfidf(q_i)}$$

where the q_i are the query terms.

Fluency: in order to improve the grammaticality of the generated sentence, we coach our ranking model to select more fluent (i.e., grammatically correct) paths in the graph. We estimate the grammaticality of generated paths ($Pr(P)$) using a language model.

Path weight: The purpose of this function is twofold: i) to generate a grammatical sentence by favoring the links between nodes (words) which appear often; and ii) to generate an informative sentence by increasing the weight of edges connecting salient nodes. For a path P with m nodes, we define the edge weight $w(n_i, n_j)$ and the path weight $W(P)$ as below:

$$w(n_i, n_j) = \frac{freq(n_i) + freq(n_j)}{\sum_{\substack{P' \in G \\ n_i, n_j \in P'}} diff(P', n_i, n_j)^{-1}}$$

$$W(P) = \frac{\sum_{i=1}^{m-1} w(n_i, n_{i+1})}{m-1}$$

where the function *diff*(P', n_i, n_j) refers to the distance between the offset positions $pos(P', n_i)$ of nodes n_i and n_j in path P' (any path in G containing n_i and n_j) and is defined as $|pos(P', n_j) - pos(P', n_i)|$.

Overal ranking score: In order to generate a query-based abstract sentence that combines the scores above, we employ a ranking model. The purpose of such a model is three-fold: i) to cover the content of query information optimally; ii) to generate a more readable and grammatical sentence; and iii) to favor strong connections between the concepts. Therefore, the final ranking score of path P is calculated over the normalized scores as:

$$Score(P) = \alpha \cdot Q(P) + \beta \cdot Pr(P) - \gamma \cdot W(P)$$

Where α, β and γ are the coefficient factors to tune the ranking score and they sum up to 1. In order to rank the graph paths, we select all the paths that contain at least one verb and rerank them using our proposed ranking function to find the best path as the summary of the original sentences in each cluster.

3 Experimental Setup

In this section, we show the evaluation results of our proposed framework and its comparison to the baselines and a state-of-the-art query-focused extractive summarization system.

3.1 Datasets

One of the challenges of this work is to find suitable conversational datasets that can be used for evaluating our query-based summarization system. Most available conversational corpora do not contain any human written summaries, or the gold standard human written summaries are generic (Carletta et al., 2005; Joty et al., 2013). In this work, we use available corpora for emails and chats for written conversations, while for spoken conversation, we employ an available corpus in multiparty meeting conversations.

Chat: to the best of our knowledge, the only publicly available chat logs with human written summaries can be downloaded from the GNUe Traffic archive (Zhou and Hovy, 2005; Uthus and Aha, 2011; Uthus and Aha, 2013). Each chat log has a human created summary in the form of a digest. Each digest summarizes IRC logs for a period and consists of few summaries over each chat log with a unique title for the associated human written summary. In this way, the title of each summary

can be counted as a phrasal query and the corresponding summary is considered as the query-based abstract of the associated chat log including only the information most relevant to the title. Therefore, we can use the human-written query-based abstract as gold standards and evaluate our system automatically. Our chat dataset consists of 66 query-based (title-based) human written summaries with their associated queries (titles) and chat logs, created from 40 original chat logs. The average number of tokens are 1840, 325 and 6 for chat logs, query-based summaries and queries, respectively.

Meeting: we use the AMI meeting corpus (Carletta et al., 2005) that consists of 140 multiparty meetings with a wide range of annotations, including generic abstractive summaries for each meeting. In order to create queries, we extract three key-phrases from generic abstractive summaries using TextRank algorithm (Mihalcea and Tarau, 2004). We use the extracted key-phrases as queries to generate query-based abstracts. Since there is no human-written query-based summary for AMI corpus, we randomly select 10 meetings and evaluate our system manually.

Email: we use BC3 (Ulrich et al., 2008), which contains 40 threads from the W3C corpus. BC3 corpus is annotated with generic human-written abstractive summaries, and it has been used in several previous works (e.g., (Joty et al., 2011)). In order to adapt this corpus to our framework, we followed the same query generation process as for the meeting dataset. Finally, we randomly select 10 emails threads and evaluate the results manually.

3.2 Baselines

We compare our approach with the following baselines:

1) Cosine-1st: we rank the utterances in the chat log based on the cosine similarity between the utterance and query. Then, we select the first uttrance as the summary;

2) Cosine-all: we rank the utterances in the chat log based on the cosine similarity between the utterance and query and then select the utterances with a cosine similarity greater than 0;

3) TextRank: a widely used graph-based ranking model for single-document sentence extraction that works by building a graph of all sentences in a document and use similarity as edges to compute the salience of sentences in the graph (Mihalcea and Tarau, 2004);

4) LexRank: another popular graph-based content selection algorithm for multi-document summarization (Erkan and Radev, 2004);

5) Biased LexRank: is a state-of-the-art query-focused summarization that uses LexRank algorithm in order to recursively retrieve additional passages that are similar to the query, as well as to the other nodes in the graph (Otterbacher et al., 2009).

Moreover, we compare our abstractive system with the first part of our framework (utterance extraction in Figure 1), which can be presented as an extractive query-based summarization system (our extractive system). We also show the results of the version we use in our pipeline (our pipeline extractive system). The only difference between the two versions is the length of the generated summaries. In our pipeline we aim at higher recall, since we later filter sentences and aggregate them to generate new abstract sentences. In contrast, in the stand alone version (extractive system) we limit the number of retrieved sentences to the desired length of the summary. We also compare the results of our full system (i.e., with tuning) with a non-optimized version when the ranking coefficients are distributed equally ($\alpha = \beta = \gamma = 0.33$). For parameters estimation, we tune all parameters (utterance selection and path ranking) exhaustively with 0.1 intervals using our development set.

For manual evaluation of query-based abstracts (meeting and email datasets), we perform a simple user study assessing the following aspects: *i) Overall quality* given a query (5-point scale)?; and *ii) Responsiveness*: how responsive is the generated summary to the query (5-point scale)? Each query-based abstract was rated by two annotators (native English speaker). Evaluators are presented with the original conversation, query and generated summary. For the manual evaluation, we only compare our full system with LexRank (LR) and Biased LexRank (Biased LR). We also ask the evaluators to select the best summary for each query and conversation, given our system generated summary and the two baselines.

To evaluate the grammaticality of our generated summaries, following common practice (Barzilay and McKeown, 2005), we randomly selected 50 sentences from original conversations and system

Models	ROUGE-1 (%)			ROUGE-2 (%)		
	Prc	Rec	F-1	Prc	Rec	F-1
Cosine-1st	71	5	8	30	3	5
Cosine-all	30	68	38	18	40	22
TextRank	25	76	34	15	44	20
LexRank	36	50	37	14	20	15
Biased LexRank	36	51	38	15	21	16
Utterance extraction (our extractive system)	34	66*	**40***†	20*†	40*	**24***†
Utterance extraction (our pipeline extractive system)	30	73*	38	19*†	44*	**24***†
Our abstractive system (without tuning)	38*	59*	**41***†	18*	27*	19*
Our abstractive system (with tuning)	40*†	56*	**42***†	20*†	25*	**22***†

Table 1: Performance of different summarization algorithms on chat logs for query-based chat summarization. Statistically significant improvements ($p < 0.01$) over the biased LexRank system are marked with *. † indicates statistical significance ($p < 0.01$) over extractive approaches (TextRank and LexRank). Systems in italics use the query in generating the summary.

generated abstracts, for each dataset. Then, we asked annotators to give one of three possible ratings for each sentence based on grammaticality: perfect (2 pts), only one mistake (1 pt) and not acceptable (0 pts), ignoring capitalization or punctuation. Each sentence was rated by two annotators. Note that each sentence was evaluated individually, so the human judges were not affected by intra-sentential problems posed by coreference and topic shifts.

3.3 Experimental Settings

For preprocessing our dataset we use OpenNLP[3] for tokenization, stemming and part-of-speech tagging. We use six randomly selected query-logs from our chat dataset (about 10% of the dataset) for tuning the coefficient parameters. We set the k parameter in our clustering phase to 10 based on the average number of sentences in the human written summaries. For our language model, we use a tri-gram smoothed language model trained using the newswire text provided in the English Gigaword corpus (Graff and Cieri, 2003). For the automatic evaluation we use the official ROUGE software with standard options and report ROUGE-1 and ROUGE-2 precision, recall and F-1 scores.

3.4 Results

3.4.1 Automatic Evaluation (Chat dataset)

Abstractive vs. Extractive: our full query-based abstractive summariztion system show statistically significant improvements over baselines and other pure extractive summarization systems for ROUGE-1[4]. This means our systems can effectively aggregate the extracted sentences and generate abstract sentences based on the query content. We can also observe that our full system produces the highest ROUGE-1 precision score among all models, which further confirms the success of this model in meeting the user information needs imposed by queries. The absolute improvement of 10% in precision for ROUGE-1 in our abstractive model over our extractive model (our pipeline) further confirms the effectiveness of our ranking method in generating the abstract sentences considering the query related information.

Our extractive query-based method beats all other extractive systems with a higher ROUGE-1 and ROUGE-2 which shows the effectiveness of our utterance extraction model in comparison with other extractive models. In other words, using our extractive model described in section 2.1, as a stand alone system, is an effective query-based extractive summarization model. We also observe that our extractive model outperforms our abstractive model for ROUGE-2 score. This can be due to word merging and word replacement choices in the word graph construction, which sometimes change or remove a word in a bigram and consequently may decrease the bigram overlap score.

Query Relevance: another interesting observation is that relying only on the cosine similarity (i.e., *cosine-all*) to measure the query relevance presents a quite strong baseline. This proves the importance of query content in our dataset and further supports the main claim of our work that a

[3] http://opennlp.apache.org/

[4] The statistical significance tests was calculated by approximate randomization, as described in (Yeh, 2000).

Dataset	Overal Quality			Responsiveness			Preference		
	Our Sys	Biased LR	LR	Our Sys	Biased LR	LR	Our Sys	Biased LR	LR
Meeting	**2.9**	2.5	2.1	**3.8**	3.2	1.8	**70%**	30%	0%
Email	**2.7**	1.8	1.7	**3.7**	3.0	1.5	**60%**	30%	10%

Table 2: Manual evaluation scores for our phrasal query abstraction system in comparison with Biased LexRank and LexRank (LR).

Dataset	Grammar		G=2		G=1		G=0	
	Orig	Sys	Orig	Sys	Orig	Sys	Orig	Sys
Chat	1.8	1.6	84%	73%	16%	24%	0%	3%
Meeting	1.5	1.3	50%	40%	50%	55%	0%	5%
Email	1.9	1.6	85%	60%	15%	35%	0%	5%

Table 3: Average rating and distribution over grammaticality scores for phrasal query abstraction system in comparison with original sentences.

good summary should express a brief and well-organized abstract that answers the user's query. Moreover, a precision of 71% for ROUGE-1 from the simple *cosine-1st* baseline confirms that some utterances contain more query relevant information in conversational discussions.

Query-based vs. Generic: the high recall and low precision in *TextRank* baseline, both for the ROUGE-1 and ROUGE-2 scores, shows the strength of the model in extracting the generic information from chat conversations while missing the query-relevant content. The *LexRank* baseline improves the results of the *TextRank* system by increasing the precision and balancing the precision and recall scores for ROUGE-1 score. We believe that this is due to the robustness of the *LexRank* method in dealing with noisy texts (chat conversations) (Erkan and Radev, 2004). In addition, the *Biased LexRank* model slightly improves the generic *LexRank* system. Considering this marginal improvement and relatively high results of pure extractive systems, we can infer that the *Biased LexRank* extracted summaries do not carry much query relevant content. In contrast, the significant improvement of our model over the extractive methods demonstrates the success of our approach in presenting the query related content in generated abstracts.

An example of a short chat log, its related query and corresponding manual and automatic summaries are shown in Example 3.

3.4.2 Manual Evaluation

Content and User Preference: Table 2 demonstrates overall quality, responsiveness (query relatedness) and user preference scores for the abstracts generated by our system and two baselines. Results indicate that our system significantly outperforms baselines in overall quality and responsiveness, for both meeting and email datasets. This confirms the validity of the results we obtained by conducting automatic evaluation over the chat dataset. We also can observe that the absolute improvements in overall quality and responsiveness for emails (0.9 and 0.7) is greater than for meetings (0.4 and 0.6). This is expected since dealing with spoken conversations is more challenging than written ones. Note that the responsiveness scores are greater than overall scores. This further proves the effectiveness of our approach in dealing with phrasal queries. We also evaluate the users' summary preferences. For both datasets (meeting and email), in majority of cases (70% and 60% respectively), the users prefer the query-based abstractive summary generated by our system.

Grammaticality: Table 3 shows grammaticality scores and distributions over the three possible scores for all datasets. The chat dataset results demonstrate the highest scores: 73% of the sentences generated by our phrasal query abstraction model are grammatically correct and 24% of the generated sentences are almost correct with only one grammatical error, while only 3% of the abstract sentences are grammatically incorrect. However, the results varies moving to other datasets. For meeting dataset, the percentage of completely grammatical sentences drops dramatically. This is due to the nature of spoken conversations which is more error prone and ungrammatical. The grammaticality score of the original sentences also proves that the sentences from meet-

Query: Trigger namespace and the self property
Chat log: A: good morning B: good morning C: good morning everyone D: good morning D: good night all F: New GNUe Traffic online F: loadsa deep metaphyisical stuff this week F: D & E discuss the meaning of 'self' ;-) E: yes, and he took the more metaphysical route, where I took the more scientific route E: I say self's meaning is derived from one's ancestry E: self's meaning is derived from how others use you E: okay, analogy extended too far, I guess :) F: is this a friends vs family debate? E: also noted that the cool part about that is if you have code that needs to happen both on a pre-insert and a pre-update - but only a few lines of the code is different E: you could have one block of trigger code that used self.action to find out why it had been called and branch accordingly. E: there was a big jump from the previous paragraph to that E: that took that out of context E: iirc, I was saying an alternative was that "self" could refer to neither the trigger's owner nor to the trigger's caller E: but to the event itself E: so self.action could be what action is being performed E: self.parent could be the trigger's owner E: and self.context could be the caller E: and self.name could be the name of the trigger F: umm, I'm afraid apparant non-sequiturs are always a hazard of doing summaries ;-) E: iow, I was presenting a 3rd alternative where self referred to something specific to the trigger F: I normally try to write around them, but not 100% successful E: I'm just convulsing my thoughts to the irc log E: for an errata next week :) E: I think convulsing is a good description F: heh
TexRank: There was a big jump from the previous paragraph to that. that took that out of context iirc, I was saying an alternative was that "self" could refer to neither the trigger's owner nor to the trigger's caller. but to the event itself. so self.action could be what action is being performed, self.parent could be the trigger's owner, and self.context could be the caller and self.name could be the name of the trigger. umm, I'm afraid apparant non-sequiturs are always a hazard of doing summaries ;-)
LexRank: good morning everyone. heh. I'm just convulsing my thoughts to the irc log good morning. Jason also noted that the cool part about that is if you have code that needs to happen both on a pre-insert and a pre-update - but only a few lines of the code is different - you could have one block of trigger code that used self.action to find out why it had been called and branch accordingly. for an errata next week :) self's meaning is derived from how others use you. I think convulsing is a good description reinhard & jcater discuss the meaning of 'self' ;-)
Biased-LexRank: good morning everyone. heh. I'm just convulsing my thoughts to the irc log. Jason also noted that the cool part about that is if you have code that needs to happen both on a pre-insert and a pre-update - but only a few lines of the code is different - you could have one block of trigger code that used self.action to find out why it had been called and branch accordingly. yes, and he took the more metaphysical route, where I took the more scientific route there was a big jump from the previous paragraph to that but to the event itself. iow, I was presenting a 3rd alternative where self referred to something specific to the trigger.
Our system: self could refer to neither the triggers owner nor caller. I was saying an alternative where self referred to something specific to the trigger. and self.name could be the name. so self.action could be what action is being performed, self.parent the triggers owner and self.context caller.
Gold: Further to, E clarified that he had suggested that "self" could refer to neither the trigger's owner nor to the trigger's caller - but to the event itself. So self.action could be what action is being performed, self.parent could be the trigger's owner, and self.context could be the caller. In other words, I was presenting a 3rd alternative where self referred to something specific to the trigger.

Example 3. Summaries generated by our system and other baselines in comparison with the human-written summary for a short chat log. Speaker information have been anonymized.

ing transcripts, although generated by humans, are not fully grammatical. In comparison with the original sentences, for all datasets, our model reports slightly lower results for the grammaticality score. Considering the fact that the abstract sentences are automatically generated and the original sentences are human-written, the grammaticality score and the percentage of fully grammatical sentences generated by our system, with higher ROUGE or quality scores in comparison with other methods, demonstrates that our system is an effective phrasal query abstraction framework for both spoken and written conversations.

4 Conclusion

We have presented an unsupervised framework for abstractive summarization of spoken and written conversations based on phrasal queries. For content selection, we propose a sentence extraction model that incorporates query relevance and content importance into the extraction process. For the generation phase, we propose a ranking strategy which selects the best path in the constructed word graph based on fluency, query relevance and content. Both automatic and manual evaluation of our model show substantial improvement over extraction-based methods, including Biased LexRank, which is considered a state-of-the-art system. Moreover, our system also yields good grammaticality score for human evaluation and achieves comparable scores with the original sentences. Our future work is four-fold. First, we are trying to improve our model by incorporating conversational features (e.g., speech acts). Second, we aim at implementing a strategy to order the clusters for generating more coherent abstracts. Third, we try to improve our generated summary by resolving coreferences and incorporating speaker information (e.g., names) in the clustering and sentence generation phases. Finally, we plan to take advantage of topic shifts to better segment the relevant parts of conversations in relation to phrasal queries.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the paper, and the NSERC Business Intelligence Network for financial support. We also would like to acknowledge the early discussions on the related topics with Frank Tompa.

References

Meni Adler, Jonathan Berant, and Ido Dagan. 2012. Entailment-based text exploration with application to the health-care domain. In *Proceedings of the ACL 2012 System Demonstrations*, ACL '12, pages 79–84, Stroudsburg, PA, USA. Association for Computational Linguistics.

Regina Barzilay and Kathleen R. McKeown. 2005. Sentence Fusion for Multidocument News Summarization. *Comput. Linguist.*, 31(3):297–328, September.

Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2011. Global Learning of Typed Entailment Rules. In *Proceedings of ACL*, Portland, OR.

Jean Carletta, Simone Ashby, Sebastien Bourban, Mike Flynn, Thomas Hain, Jaroslav Kadlec, Vasilis Karaiskos, Wessel Kraaij, Melissa Kronenthal, Guillaume Lathoud, Mike Lincoln, Agnes Lisowska, and Mccowan Wilfried Post Dennis Reidsma. 2005. The AMI meeting corpus: A pre-announcement. In *Proc. MLMI*, pages 28–39.

I. Dagan and O. Glickman. 2004. Probabilistic Textual Entailment: Generic applied modeling of language variability. In *PASCAL Workshop on Learning Methods for Text Understanding and Mining*.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.*, 22(1):457–479, December.

Katja Filippova. 2010. Multi-sentence compression: finding shortest paths in word graphs. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 322–330, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 340–348, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dan Gillick, Korbinian Riedhammer, Benoit Favre, and Dilek Hakkani-tr. 2009. A global optimization framework for meeting summarization. In *Proc. IEEE ICASSP*, pages 4769–4772.

Julio Gonzalo, Felisa Verdejo, Irina Chugur, and Juan M. Cigarrn. 1998. Indexing with wordnet synsets can improve text retrieval. *CoRR*.

David Graff and Christopher Cieri. 2003. English Gigaword Corpus. Technical report, Linguistic Data Consortium, Philadelphia.

Surabhi Gupta, Ani Nenkova, and Dan Jurafsky. 2007. Measuring importance and query relevance in topic-focused multi-document summarization. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 193–196, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lisa Hunemark. 2010. Query expansion using search logs and WordNet. Technical report, Uppsala University, mar. Masters thesis in Computational Linguistics.

Quentin Jones, Gilad Ravid, and Sheizaf Rafaeli. 2004. Information overload and the message dynamics of online interaction spaces: A theoretical model and empirical exploration. *Info. Sys. Research*, 15(2):194–210, June.

Shafiq Joty, Gabriel Murray, and Raymond T. Ng. 2011. Supervised topic segmentation of email conversations. In *ICWSM11*. AAAI.

Shafiq R. Joty, Giuseppe Carenini, and Raymond T. Ng. 2013. Topic segmentation and labeling in asynchronous conversations. *J. Artif. Intell. Res. (JAIR)*, 47:521–573.

Chin-Yew Lin and Eduard Hovy. 2000. The automated acquisition of topic signatures for text summarization. In *Proc. Of the COLING Conference*, pages 495–501.

Fei Liu and Yang Liu. 2009. From extractive to abstractive meeting summaries: can it be done by sentence compression? In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, ACLShort '09, pages 261–264, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yashar Mehdad, Giuseppe Carenini, and Raymond NG T. 2013a. Towards Topic Labeling with Phrase Entailment and Aggregation. In *Proceedings of NAACL 2013*, pages 179–189, Atlanta, USA, June. Association for Computational Linguistics.

Yashar Mehdad, Giuseppe Carenini, Frank Tompa, and Raymond T. NG. 2013b. Abstractive meeting summarization with entailment and fusion. In *Proceedings of the 14th European Workshop on Natural Language Generation*, pages 136–146, Sofia, Bulgaria, August. Association for Computational Linguistics.

Weiyi Meng and Clement T. Yu. 2010. *Advanced Metasearch Engine Technology*. Synthesis Lectures on Data Management. Morgan and Claypool Publishers.

R. Mihalcea and P. Tarau. 2004. TextRank: Bringing order into texts. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, July.

Gabriel Murray, Giuseppe Carenini, and Raymond Ng. 2010. Generating and validating abstracts of meeting conversations: a user study. In *Proceedings of*

the 6th International Natural Language Generation Conference, INLG '10, pages 105–113, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jahna Otterbacher, Gnes Erkan, and Dragomir R. Radev. 2009. Biased lexrank: Passage retrieval using random walks with question-based priors. *Inf. Process. Manage.*, 45(1):42–54.

Tetsuya Sakai and Karen Sparck-Jones. 2001. Generic summaries for indexing in information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 190–198, New York, NY, USA. ACM.

J. Ulrich, G. Murray, and G. Carenini. 2008. A publicly available annotated corpus for supervised email summarization. In *AAAI08 EMAIL Workshop*, Chicago, USA. AAAI.

David C. Uthus and David W. Aha. 2011. Plans toward automated chat summarization. In *Proceedings of the Workshop on Automatic Summarization for Different Genres, Media, and Languages*, WASDGML '11, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

David C. Uthus and David W. Aha. 2013. The ubuntu chat corpus for multiparticipant chat analysis. In *AAAI Spring Symposium: Analyzing Microtext*.

Lu Wang and Claire Cardie. 2013. Domain-independent abstract generation for focused meeting summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1395–1405, Sofia, Bulgaria, August. Association for Computational Linguistics.

Lu Wang, Hema Raghavan, Vittorio Castelli, Radu Florian, and Claire Cardie. 2013. A sentence compression based framework to query-focused multi-document summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1384–1394, Sofia, Bulgaria, August. Association for Computational Linguistics.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th Conference on Computational Linguistics - Volume 2*, COLING '00, pages 947–953. Association for Computational Linguistics.

Liang Zhou and Eduard Hovy. 2005. Digesting virtual "geek" culture: The summarization of technical internet relay chats. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 298–305, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Comparing Multi-label Classification with Reinforcement Learning for Summarisation of Time-series Data

Dimitra Gkatzia, Helen Hastie, and Oliver Lemon
School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh
{dg106, h.hastie, o.lemon}@hw.ac.uk

Abstract

We present a novel approach for automatic report generation from time-series data, in the context of student feedback generation. Our proposed methodology treats content selection as a multi-label (ML) classification problem, which takes as input time-series data and outputs a set of templates, while capturing the dependencies between selected templates. We show that this method generates output closer to the feedback that lecturers actually generated, achieving 3.5% higher accuracy and 15% higher F-score than multiple simple classifiers that keep a history of selected templates. Furthermore, we compare a ML classifier with a Reinforcement Learning (RL) approach in simulation and using ratings from real student users. We show that the different methods have different benefits, with ML being more accurate for predicting what was seen in the training data, whereas RL is more exploratory and slightly preferred by the students.

1 Introduction

Summarisation of time-series data refers to the task of automatically generating text from variables whose values change over time. We consider the task of automatically generating feedback summaries for students describing their performance during the lab of a Computer Science module over the semester. Students' learning can be influenced by many variables, such as difficulty of the material (Person et al., 1995), other deadlines (Craig et al., 2004), attendance in lectures (Ames, 1992), etc. These variables have two important qualities. Firstly, they change over time, and secondly they can be dependent on or independent of each other. Therefore, when generating feedback, we need to take into account all variables simultaneously in order to capture potential dependencies and provide more effective and useful feedback that is relevant to the students.

In this work, we concentrate on content selection which is the task of choosing what to say, i.e. what information is to be included in a report (Reiter and Dale, 2000). Content selection decisions based on trends in time-series data determine the selection of the useful and important variables, which we refer to here as *factors*, that should be conveyed in a summary. The decisions of factor selection can be influenced by other factors that their values are correlated with; can be based on the appearance or absence of other factors in the summary; and can be based on the factors' behaviour over time. Moreover, some factors may have to be discussed together in order to achieve some communicative goal, for instance, a teacher might want to refer to student's marks as a motivation for increasing the number of hours studied.

We frame content selection as a simple classification task: given a set of time-series data, decide for each template whether it should be included in a summary or not. In this paper, with the term 'template' we refer to a quadruple consisting of an *id*, a *factor* (bottom left of Table 1), a *reference type* (trend, weeks, average, other) and *surface text*. However, simple classification assumes that the templates are independent of each other, thus the decision for each template is taken in isolation from the others, which is not appropriate for our domain. In order to capture the dependencies in the context, multiple simple classifiers can make the decisions for each template iteratively. After each iteration, the feature space grows by 1 feature, in order to include the history of the previous template decisions. Here, we propose an alternative method that tackles the challenge of interdependent data by using multi-label (ML) classification, which is efficient in taking data dependencies

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1231–1240,
Baltimore, Maryland, USA, June 23-25 2014.

Raw Data

factors	week 2	week 3	...	week 10
marks	5	4	...	5
hours_studied	1	2	...	3
...	...	...	...	...

Trends from Data

factors	trend
(1) marks (M)	trend_other
(2) hours_studied (HS)	trend_increasing
(3) understandability (Und)	trend_decreasing
(4) difficulty (Diff)	trend_decreasing
(5) deadlines (DL)	trend_increasing
(6) health_issues (HI)	trend_other
(7) personal_issues (PI)	trend_decreasing
(8) lectures_attended (LA)	trend_other
(9) revision (R)	trend_decreasing

Summary

Your overall performance **was excellent** during the semester. Keep up the good work and maybe try some more challenging exercises. Your attendance was **varying** over the semester. Have a think about how to use time in lectures to improve your understanding of the material. You spent **2 hours studying the lecture material on average**. You should dedicate more time to study. You seem to find the material **easier to understand compared to the beginning of the semester**. Keep up the good work! You revised **part of** the learning material. Have a think whether revising has improved your performance.

Table 1: The table on the top left shows an example of the time-series raw data for feedback generation. The table on the bottom left shows an example of described trends. The box on the right presents a target summary (target summaries have been constructed by teaching staff).

into account and generating a set of labels (in our case templates) simultaneously (Tsoumakas et al., 2010). ML classification requires no history, i.e. does not keep track of previous decisions, and thus has a smaller feature space.

Our contributions to the field are as follows: we present a novel and efficient method for tackling the challenge of content selection using a ML classification approach; we applied this method to the domain of feedback summarisation; we present a comparison with an optimisation technique (Reinforcement Learning), and we discuss the similarities and differences between the two methods.

In the next section, we refer to the related work on Natural Language Generation from time-series data and on Content Selection. In Section 4.2, we describe our approach and we carry out a comparison with simple classification methods. In Section 5, we present the evaluation setup and in Section 6 we discuss the results, obtained in simulation and with real students. Finally, in Section 8, directions for future work are discussed.

2 Related Work

Natural Language Generation from time-series data has been investigated for various tasks such as weather forecast generation (Belz and Kow, 2010; Angeli et al., 2010; Sripada et al., 2004), report generation from clinical data (Hunter et al., 2011; Gatt et al., 2009), narrative to assist children with communication needs (Black et al., 2010) and audiovisual debrief generation from sensor data from Autonomous Underwater Vehicles missions (Johnson and Lane, 2011).

The important tasks of time-series data summarisation systems are *content selection* (what to say), *surface realisation* (how to say it) and *information presentation* (Document Planning, Ordering, etc.). In this work, we concentrate on content selection. Previous methods for content selection include Reinforcement Learning (Rieser et al., 2010); multi-objective optimisation (Gkatzia et al., 2014); Gricean Maxims (Sripada et al., 2003); Integer Linear Programming (Lampouras and Androutsopoulos, 2013); collective content selection (Barzilay and Lapata, 2004); interest scores assigned to content (Androutsopoulos et al., 2013); a combination of statistical and template-based approaches to NLG (Kondadadi et al., 2013); statistical acquisition of rules (Duboue and McKeown, 2003) and the Hidden Markov model approach for Content Selection and ordering (Barzilay and Lee, 2004).

Collective content selection (Barzilay and Lapata, 2004) is similar to our proposed method in that it is a classification task that predicts the templates from the same instance simultaneously. The difference between the two methods lies in that the

collective content selection requires the consideration of an individual preference score (which is defined as the preference of the entity to be selected or omitted, and it is based on the values of entity attributes and is computed using a boosting algorithm) and the identification of links between the entities with similar labels. In contrast, ML classification does not need the computation of links between the data and the templates. ML classification can also apply to other problems whose features are correlated, such as text classification (Madjarov et al., 2012), when an aligned dataset is provided.

ML classification algorithms have been divided into three categories: algorithm adaptation methods, problem transformation and ensemble methods (Tsoumakas and Katakis, 2007; Madjarov et al., 2012). Algorithm adaptation approaches (Tsoumakas et al., 2010) extend simple classification methods to handle ML data. For example, the k-nearest neighbour algorithm is extended to ML-kNN by Zhang and Zhou (2007). ML-kNN identifies for each new instance its k nearest neighbours in the training set and then it predicts the label set by utilising the maximum a posteriori principle according to statistical information derived from the label sets of the k neighbours. Problem transformation approaches (Tsoumakas and Katakis, 2007) transform the ML classification task into one or more simple classification tasks. Ensemble methods (Tsoumakas et al., 2010) are algorithms that use ensembles to perform ML learning and they are based on problem transformation or algorithm adaptation methods. In this paper, we applied RAkEL (Random k-labelsets) (Tsoumakas et al., 2010): an ensemble problem transformation method, which constructs an ensemble of simple-label classifiers, where each one deals with a random subset of the labels.

Finally, our domain for feedback generation is motivated by previous studies (Law et al., 2005; van den Meulen et al., 2010) who show that text summaries are more effective in decision making than graphs therefore it is advantageous to provide a summary over showing users the raw data graphically. In addition, feedback summarisation from time-series data can be applied to the field of Intelligent Tutoring Systems (Gross et al., 2012).

3 Data

The dataset consists of 37 instances referring to the activities of 26 students. For a few students there is more than 1 instance. An example of one such instance is presented in Table 1. Each instance includes time-series information about the student's learning habits and the selected templates that lecturers used to provide feedback to this student. The time-series information includes for each week of the semester: (1) the marks achieved at the lab; (2) the hours that the student spent studying; (3) the understandability of the material; (4) the difficulty of the lab exercises as assessed by the student; (5) the number of other deadlines that the student had that week; (6) health issues; (7) personal issues; (8) the number of lectures attended; and (9) the amount of revision that the student had performed. The templates describe these factors in four different ways:

1. <**trend**>**:** referring to the trend of a factor over the semester (e.g. "Your performance *was increasing*..."),
2. <**weeks**>**:** explicitly describing the factor value at specific weeks (e.g. "In *weeks 2, 3 and 9*..."),
3. <**average**>**:** considering the average of a factor value (e.g. "You dedicated *1.5 hours studying on average*..."), and
4. <**other**>**:** mentioning other relevant information (e.g. "*Revising material will improve your performance*").

For the corpus creation, 11 lecturers selected the content to be conveyed in a summary, given the set of raw data (Gkatzia et al., 2013). As a result, for the same student there are various summaries provided by the different experts. This characteristic of the dataset, that each instance is associated with more than one solution, additionally motivates the use of multi-label classification, which is concerned with learning from examples, where each example is associated with multiple labels.

Our analysis of the dataset showed that there are significant correlations between the factors, for example, the number of lectures attended (LA) correlates with the student's understanding of the material (Und), see Table 2. As we will discuss further in Section 5.1, content decisions are influenced by the previously generated content, for example, if the lecturer has previously mentioned health_issues, mentioning hours_studied has a high probability of also being mentioned.

Factor	(1) M	(2) HS	(3) Und	(4) Diff	(5) DL	(6) HI	(7) PI	(8) LA	(9) R
(1) M	1*	0.52*	0.44*	-0.53*	-0.31	-0.30	-0.36*	0.44*	0.16
(2) HS	0.52*	1*	0.23	-0.09	-0.11	0.11	-0.29	0.32	0.47*
(3) Und	0.44*	0.23	1*	-0.54*	0.03	-0.26	0.12	0.60*	0.32
(4) Diff	-0.53*	-0.09	-0.54*	1*	0.16	-0.06	0.03	-0.19	0.14
(5) DL	-0.31	-0.11	0.03	0.16	1*	0.26	0.24	-0.44*	0.14
(6) HI	-0.30	-0.11	-0.26	-0.06	0.26	1*	0.27	-0.50*	0.15
(7) PI	-0.36*	-0.29	0.12	0.03	0.24	0.27	1*	-0.46*	0.34*
(8) LA	0.44*	0.32	0.60*	-0.19	-0.44*	-0.50*	-0.46*	1*	-0.12
(9) R	0.16	0.47*	0.03	0.14	0.14	0.15	0.34*	-0.12	1*

Table 2: The table presents the Pearson's correlation coefficients of the factors (* means $p<0.05$).

4 Methodology

In this section, the content selection task and the suggested multi-label classification approach are presented. The development and evaluation of the time-series generation system follows the following pipeline (Gkatzia et al., 2013):

1. Time-Series data collection from students
2. Template construction by Learning and Teaching (L&T) expert
3. Feedback summaries constructed by lecturers; random summaries rated by lecturers
4. Development of time-series generation systems (Section 4.2, Section 5.3): ML system, RL system, Rule-based and Random system
5. Evaluation: (Section 5)
   - Offline evaluation (Accuracy and Reward)
   - Online evaluation (Subjective Ratings)

4.1 The Content Selection Task

Our learning task is formed as follows: given a set of 9 time-series factors, select the content that is most appropriate to be included in a summary. Content is regarded as labels (each template represents a label) and thus the task can be thought of as a classification problem. As mentioned, there are 4 ways to refer to a factor: (1) describing the trend, (2) describing what happened in every time stamp, (3) mentioning the average and (4) making another general statement. Overall, for all factors there are 29 different templates[1]. An example of the input data is shown in Table 1. There are two decisions that need to be made: (1) whether to talk about a factor and (2) in which way to refer to it. Instead of dealing with this task in a hierarchical way, where the algorithm will first learn whether to talk about a factor and then to decide how to refer to it, we transformed the task in order to reduce the learning steps. Therefore, classification can reduce the decision workload by deciding either in which way to talk about it, or not to talk about a factor at all.

[1]There are fewer than 36 templates, because for some factors there are less than 4 possible ways of referring to them.

4.2 The Multi-label Classification Approach

Traditional single-label classification is the task of identifying which label one new observation is associated with, by choosing from a set of labels L (Tsoumakas et al., 2010). Multi-label classification is the task of associating an observation with a set of labels $Y \subseteq L$ (Tsoumakas et al., 2010).

One set of factor values can result in various sets of templates as interpreted by the different experts. A ML classifier is able to make decisions for all templates simultaneously and capture these differences. The RAndom k-labELsets (RAkEL) (Tsoumakas et al., 2010) was applied in order to perform ML classification. RAkEL is based on Label Powerset (LP), a problem transformation method (Tsoumakas et al., 2010). LP benefits from taking into consideration label correlations, but does not perform well when trained with few examples as in our case (Tsoumakas et al., 2010). RAkEL overcomes this limitation by constructing a set of LP classifiers, which are trained with different random subsets of the set of labels (Tsoumakas et al., 2010).

The LP method transforms the ML task, into one single-label multi-class classification task, where the possible set of predicted variables for the transformed class is the powerset of labels present in the original dataset. For instance, the set of labels $L = \{temp_0, temp_1, ...temp_{28}\}$ could be transformed to $\{temp_{0,1,2}, temp_{28,3,17,}...\}$. This algorithm does not perform well when considering a large number of labels, due to the fact that the label space grows exponentially (Tsoumakas

Classifier	Accuracy (10-fold)	Precision	Recall	F score
Decision Tree (no history)	*75.95%	67.56	75.96	67.87
Decision Tree (with predicted history)	**73.43%	65.49	72.05	70.95
Decision Tree (with real history)	**78.09%	74.51	78.11	75.54
Majority-class (single label)	**72.02%	61.73	77.37	68.21
RAkEL (multi-label) (no history)	76.95%	**85.08**	**85.94**	**85.50**

Table 3: Average, precision, recall and F-score of the different classification methods (T-test, * denotes significance with $p<0.05$ and ** significance with $p<0.01$, when comparing each result to RAkEL).

et al., 2010). RAkEL tackles this problem by constructing an ensemble of LP classifiers and training each one on a different random subset of the set of labels (Tsoumakas et al., 2010).

4.2.1 The Production Phase of RAkEL

The algorithm was implemented using the MULAN Open Source Java library (Tsoumakas et al., 2011), which is based on WEKA (Witten and Frank, 2005). The algorithm works in two phases:

1. the production of an ensemble of LP algorithms, and
2. the combination of the LP algorithms.

RAkEL takes as input the following parameters: (1) the numbers of iterations m (which is developer specified and denotes the number of models that the algorithm will produce), (2) the size of labelset k (which is also developer specified), (3) the set of labels L, and (4) the training set D. During the initial phase it outputs an ensemble of LP classifiers and the corresponding k-labelsets. A pseudocode for the production phase is shown below:

Algorithm 1 RAkEL production phase

```
1: Input: iterations m, k labelsets,
          labels L, training data D

2: for i=0 to m
3:       Select random k-labelset from L
4:       Train an LP on D
5:       Add LP to ensemble
6: end for

7: Output: the ensemble of LPs
       with corresponding k-labelsets
```

4.2.2 The Combination Phase

During the combination phase, the algorithm takes as input the results of the production phase, i.e. the ensemble of LPs with the corresponding k-labelsets, the set of labels L, and the new instance x and it outputs the result vector of predicted labels for instance x. During run time, RAkEL estimates the average decision for each label in L and if the average is greater than a threshold t (determined by the developer) it includes the label in the predicted labelset. We used the standard parameter values of t, k and m (t = 0.5, k = 3 and m equals to 58 (2*29 templates)). In future, we could perform parameter optimisation by using a technique similar to (Gabsdil and Lemon, 2004).

5 Evaluation

Firstly, we performed a preliminary evaluation on classification methods, comparing our proposed ML classification with multiple iterated classification approaches. The summaries generated by the ML classification system are then compared with the output of a RL system and two baseline systems in simulation and with real students.

5.1 Comparison with Simple Classification

We compared the RAkEL algorithm with single-label (SL) classification. Different SL classifiers were trained using WEKA: JRip, Decision Trees, Naive Bayes, k-nearest neighbour, logistic regression, multi-layer perceptron and support vector machines. It was found out that Decision Trees achieved on average 3% higher accuracy. We, therefore, went on to use Decision Trees that use generation history in three ways.

Firstly, for **Decision Tree (no history)**, 29 decision-tree classifiers were trained, one for each template. The input of these classifiers were the 9 factors and each classifier was trained in order to decide whether to include a specific template or not. This method did not take into account other selected templates – it was only based on the time-series data.

Secondly, for **Decision Tree (with predicted history)**, 29 classifiers were also trained, but this time the input included the previous decisions made by the previous classifiers (i.e. the history)

as well as the set of time-series data in order to emulate the dependencies in the dataset. For instance, classifier n was trained using the data from the 9 factors and the template decisions for templates 0 to $n-1$.

Thirdly, for **Decision Tree (with real history)**, the real, expert values were used rather than the predicted ones in the history. The above-mentioned classifiers are compared with, the **Majority-class (single label)** baseline, which labels each instance with the most frequent template.

The accuracy, the weighted precision, the weighted recall, and the weighted F-score of the classifiers are shown in Table 3. It was found that in 10-fold cross validation RAkEL performs significantly better in all these automatic measures (accuracy = 76.95%, F-score = 85.50%). Remarkably, ML achieves more than 10% higher F-score than the other methods (Table 3). The average accuracy of the single-label classifiers is 75.95% (10-fold validation), compared to 73.43% of classification with history. The reduced accuracy of the classification with predicted history is due to the error in the predicted values. In this method, at every step, the predicted outcome was used including the incorrect decisions that the classifier made. The upper-bound accuracy is 78.09% calculated by using the expert previous decisions and not the potentially erroneous predicted decisions. This result is indicative of the significance of the relations between the factors showing that the predicted decisions are dependent due to existing correlations as discussed in Section 1, therefore the system should not take these decisions independently. ML classification performs better because it does take into account these correlations and dependencies in the data.

5.2 The Reinforcement Learning System

Reinforcement Learning (RL) is a machine learning technique that defines how an agent learns to take optimal actions so as to maximise a cumulative reward (Sutton and Barto, 1998). Content selection is seen as a Markov Decision problem and the goal of the agent is to learn to take the sequence of actions that leads to optimal content selection. The Temporal Difference learning method was used to train an agent for content selection.

Actions and States: The state consists of the time-series data and the selected templates. In order to explore the state space the agent selects a factor (e.g. marks, deadlines etc.) and then decides whether to talk about it or not.

Reward Function: The reward function reflects the lecturers' preferences on summaries and is derived through linear regression analysis of a dataset containing lecturer constructed summaries and ratings of randomly generated summaries. Specifically, it is the following cumulative multivariate function:

$$Reward = a + \sum_{i=1}^{n} b_i * x_i + c * length$$

where $X = \{x_1, x_2, ..., x_n\}$ describes the combinations of the data trends observed in the time-series data and a particular template. a, b and c are the regression coefficients, and their values vary from -99 to 221. The value of x_i is given by the function:

$$x_i = \begin{cases} 1, & \text{the combination of a factor trend} \\ & \text{and a template type is included} \\ & \text{in a summary} \\ 0, & \text{if not.} \end{cases}$$

The RL system differs from the classification system in the way it performs content selection. In the training phase, the agent selects a factor and then decides whether to talk about it or not. If the agent decides to refer to a factor, the template is selected in a deterministic way, i.e. from the available templates it selects the template that results in higher expected cumulative future reward.

5.3 The Baseline Systems

We compared the ML system and the RL system with two baselines described below by measuring the accuracy of their outputs, the reward achieved by the reward function used for the RL system, and finally we also performed evaluation with student users. In order to reduce the confounding variables, we kept the ordering of content in all systems the same, by adopting the ordering of the rule-based system. The baselines are as follows:

1. Rule-based System: generates summaries based on Content Selection rules derived by working with a L&T expert and a student (Gkatzia et al., 2013).

2. Random System: initially, selects a factor randomly and then selects a template randomly, until it makes decisions for all factors.

Time-Series Summarisation Systems	Accuracy	Reward	Rating Mode (mean)	Data Source
Multi-label Classification	**85%**	65.4	7 (6.24)	Lecturers' constructed summaries
Reinforcement Learning	**66%	**243.82**	**8 (6.54)**	Lecturers' ratings & summaries
Rule-based	**65%	107.77	7, 8 (5.86)	L&T expert
Random	**45.2%	43.29	*2 (*4.37)	Random

Table 4: Accuracy, average rewards (based on lecturers' preferences) and averages of the means of the student ratings. Accuracy significance (Z-test) with RAkEL at $p<0.05$ is indicated as * and at $p<0.01$ as **. Student ratings significance (Mann Whitney U test) with RAkEL at $p<0.05$ is indicated as *.

6 Results

Each of the four systems described above generated 26 feedback summaries corresponding to the 26 student profiles. These summaries were evaluated in simulation and with real student users.

6.1 Results in Simulation

Table 4 presents the accuracy, reward, and mode of student rating of each algorithm when used to generate the 26 summaries. Accuracy was estimated as the proportion of the correctly classified templates to the population of templates. In order to have a more objective view on the results, the score achieved by each algorithm using the reward function was also calculated. ML classification achieved significantly higher accuracy, which was expected as it is a supervised learning method. The rule-based system and the RL system have lower accuracy compared to the ML system. There is evidently a mismatch between the rules and the test-set; the content selection rules are based on heuristics provided by a L&T Expert rather than by the same pool of lecturers that created the test-set. On the contrary, the RL is trained to optimise the selected content and not to replicate the existing lecturer summaries, hence there is a difference in accuracy.

Accuracy measures how similar the generated output is to the gold standard, whereas the reward function calculates a score regarding how good the output is, given an objective function. RL is trained to optimise for this function, and therefore it achieves higher reward, whereas ML is trained to learn by examples, therefore it produces output closer to the gold standard (lecturer's produced summaries). RL uses exploration and exploitation to discover combinations of content that result in higher reward. The reward represents predicted ratings that lecturers would give to the summary. The reward for the lecturers' produced summaries is 124.62 and for the ML method is 107.77. The ML classification system performed worse than this gold standard in terms of reward, which is expected given the error in predictions (supervised methods learn to reproduce the gold standard). Moreover, each decision is rewarded with a different value as some combinations of factors and templates have greater or negative regression coefficients. For instance, the combination of the factors "deadlines" and the template that corresponds to <weeks> is rewarded with 57. On the other hand, when mentioning the <average> difficulty the summary is "punished" with -81 (see description of the reward function in Section 5.2). Consequently, a single poor decision in the ML classification can result in much less reward.

6.2 Subjective Results with Students

37 first year computer science students participated in the study. Each participant was shown a graphical representation of the time-series data of one student and four different summaries generated by the four systems (see Figure 1). The order of the presented summaries was randomised. They were asked to rate each feedback summary on a 10-point rating scale in response to the following statement: "Imagine you are the following student. How would you evaluate the following feedback summaries from 1 to 10?", where 10 corresponds to the most preferred summary and 1 to the least preferred.

The difference in ratings between the ML classification system, the RL system and the Rule-based system is not significant (see Mode (mean) in Table 4, $p>0.05$). However, there is a trend towards the RL system. The classification method reduces the generation steps, by making the decision of the factor selection and the template selection jointly. Moreover, the training time for the classification method is faster (a couple of seconds compared to over an hour). Finally, the student

Figure 1: The Figure show the evaluation setup. Students were presenting with the data in a graphical way and then they were asked to evaluate each summary in a 10-point Rating scale. Summaries displayed from left to right: ML system, RL, rule-based and random.

significantly prefer all the systems over the random.

7 Summary

We have shown that ML classification for summarisation of our time-series data has an accuracy of 76.95% and that this approach significantly outperforms other classification methods as it is able to capture dependencies in the data when making content selection decisions. ML classification was also directly compared to a RL method. It was found that although ML classification is almost 20% more accurate than RL, both methods perform comparably when rated by humans. This may be due to the fact that the RL optimisation method is able to provide more varied responses over time rather than just emulating the training data as with standard supervised learning approaches. Foster (2008) found similar results when performing a study on generation of emphatic facial displays. A previous study by Belz and Reiter (2006) has demonstrated that automatic metrics can correlate highly with human ratings if the training dataset is of high quality. In our study, the human ratings correlate well to the average scores achieved by the reward function. However, the human ratings do not correlate well to the accuracy scores. It is interesting that the two methods that score differently on various automatic metrics, such as accuracy, reward, precision, recall and F-score, are evaluated similarly by users.

The comparison shows that each method can serve different goals. Multi-label classification generates output closer to gold standard whereas RL can optimise the output according to a reward function. ML classification could be used when the goal of the generation is to replicate phenomena seen in the dataset, because it achieves high accuracy, precision and recall. However, optimisation methods can be more flexible, provide more varied output and can be trained for different goals, e.g. for capturing preferences of different users.

8 Future Work

For this initial experiment, we evaluated with students and not with lecturers, since the students are the recipients of feedback. In future, we plan to evaluate with students' own data under real circumstances as well as with ratings from lecturers. Moreover, we plan to utilise the results from this student evaluation in order to train an optimisation algorithm to perform summarisation according to students' preferences. In this case, optimisation would be the preferred method as it would not be appropriate to collect gold standard data from students. In fact, it would be of interest to investigate multi-objective optimisation techniques that can balance the needs of the lecturers to convey important content to the satisfaction of students.

9 Acknowledgements

The research leading to this work has received funding from the EC's FP7 programme: (FP7/2011-14) under grant agreement no. 248765 (Help4Mood).

References

Carole Ames. 1992. Classrooms: Goals, structures, and student motivation. *Journal of Educational Psychology*, 84(3):261–71.

Ion Androutsopoulos, Gerasimos Lampouras, and Dimitrios Galanis. 2013. Generating natural language descriptions from owl ontologies: the natural owl system. *Atrificial Intelligence Research*, 48:671–715.

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Regina Barzilay and Mirella Lapata. 2004. Collective content selection for concept-to-text generation. In *Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT-EMNLP)*.

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*.

Anja Belz and Eric Kow. 2010. Extracting parallel fragments from comparable corpora for data-to-text generation. In *6th International Natural Language Generation Conference (INLG)*.

Anja Belz and Ehud Reiter. 2006. Comparing automatic and human evaluation of nlg systems. In *11th Conference of the European Chapter of the Association for Computational Linguistics (ACL)*.

Rolf Black, Joe Reddington, Ehud Reiter, Nava Tintarev, and Annalu Waller. 2010. Using NLG and sensors to support personal narrative for children with complex communication needs. In *NAACL HLT 2010 Workshop on Speech and Language Processing for Assistive Technologies*.

Scotty D. Craig, Arthur C. Graesser, Jeremiah Sullins, and Barry Gholson. 2004. Affect and learning: an exploratory look into the role of affect in learning with autotutor. *Journal of Educational Media*, 29:241–250.

Pable Duboue and K.R. McKeown. 2003. Statistical acquisition of content selection rules for natural language generation. In *Conference on Human Language Technology and Empirical Methods in Natural Language Processing (EMNLP)*.

Mary Ellen Foster. 2008. Automated metrics that agree with human judgements on generated output for an embodied conversational agent. In *5th International Natural Language Generation Conference (INLG)*.

Malte Gabsdil and Oliver Lemon. 2004. Combining acoustic and pragmatic features to predict recognition performance in spoken dialogue systems. In *42nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Albert Gatt, Francois Portet, Ehud Reiter, James Hunter, Saad Mahamood, Wendy Moncur, and Somayajulu Sripada. 2009. From data to text in the neonatal intensive care unit: Using NLG technology for decision support and information management. *AI Communications*, 22: 153-186.

Dimitra Gkatzia, Helen Hastie, Srinivasan Janarthanam, and Oliver Lemon. 2013. Generating student feedback from time-series data using Reinforcement Learning. In *14th European Workshop in Natural Language Generation (ENLG)*.

Dimitra Gkatzia, Helen Hastie, and Oliver Lemon. 2014. Finding Middle Ground? Multi-objective Natural Language Generation from time-series data. In *14th Conference of the European Chapter of the Association for Computational Linguistics (EACL) (to appear)*.

Sebastian Gross, Bassam Mokbel, Barbara Hammer, and Niels Pinkwart. 2012. Feedback provision strategies in intelligent tutoring systems based on clustered solution spaces. In J. Desel, J. M. Haake, and C. Spannagel, editors, *Tagungsband der 10. e-Learning Fachtagung Informatik (DeLFI)*, number P-207 in GI Lecture Notes in Informatics, pages 27–38. GI.

Jim Hunter, Yvonne Freer, Albert Gatt, Yaji Sripada, Cindy Sykes, and D Westwater. 2011. Bt-nurse: Computer generation of natural language shift summaries from complex heterogeneous medical data. *American Medical Informatics Association*, 18:621-624.

Nicholas Johnson and David Lane. 2011. Narrative monologue as a first step towards advanced mission debrief for AUV operator situational awareness. In *15th International Conference on Advanced Robotics*.

Ravi Kondadadi, Blake Howald, and Frank Schilder. 2013. A statistical nlg framework for aggregated planning and realization. In *51st Annual Meeting of the Association for Computational Linguistics (ACL)*.

Gerasimos Lampouras and Ion Androutsopoulos. 2013. Using integer linear programming in concept-to-text generation to produce more compact texts. In *51st Annual Meeting of the Association for Computational Linguistics (ACL)*.

Anna S. Law, Yvonne Freer, Jim Hunter, Robert H. Logie, Neil McIntosh, and John Quinn. 2005. A comparison of graphical and textual presentations of time series data to support medical decision making in the neonatal intensive care unit. *Journal of Clinical Monitoring and Computing*, pages 19: 183–194.

Gjorgji Madjarov, Dragi Kocev, Dejan Gjorgjevikj, and Saso Dzeroski. 2012. An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, 45(9):3084–3104.

Natalie K. Person, Roger J. Kreuz, Rolf A. Zwaan, and Arthur C. Graesser. 1995. Pragmatics and pedagogy: Conversational rules and politeness strategies may inhibit effective tutoring. Journal of Cognition and Instruction, 13(2):161-188.

Ehud Reiter and Robert Dale. 2000. Building natural language generation systems. Cambridge University Press.

Verena Rieser, Oliver Lemon, and Xingkun Liu. 2010. Optimising information presentation for spoken dialogue systems. In *48th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Somayajulu Sripada, Ehud Reiter, Jim Hunter, and Jin Yu. 2003. Generating english summaries of time series data using the gricean maxims. In *9th ACM international conference on Knowledge discovery and data mining (SIGKDD)*.

Somayajulu Sripada, Ehud Reiter, I Davy, and K Nilssen. 2004. Lessons from deploying NLG technology for marine weather forecast text generation. In *PAIS session of ECAI-2004:760-764*.

Richart Sutton and Andrew Barto. 1998. Reinforcement learning. MIT Press.

Grigorios Tsoumakas and Ioannis Katakis. 2007. Multi-label classification: An overview. *International Journal Data Warehousing and Mining*, 3(3):1–13.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2010. Random k-labelsets for multi-label classification. *IEEE Transactions on Knowledge and Data Engineering*, 99(1):1079–1089.

Grigorios Tsoumakas, Eleftherios Spyromitros-Xioufis, Josef Vilcek, and Ioannis Vlahavas. 2011. Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, 12(1):2411–2414.

Marian van den Meulen, Robert Logie, Yvonne Freer, Cindy Sykes, Neil McIntosh, and Jim Hunter. 2010. When a graph is poorer than 100 words: A comparison of computerised natural language generation, human generated descriptions and graphical displays in neonatal intensive care. In *Applied Cognitive Psychology, 24: 77-89*.

Ian Witten and Eibe Frank. 2005. Data mining: Practical machine learning tools and techniques. Morgan Kaufmann Publishers.

Min-Ling Zhang and Zhi-Hua Zhou. 2007. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048.

Approximation Strategies for Multi-Structure Sentence Compression

Kapil Thadani
Department of Computer Science
Columbia University
New York, NY 10025, USA
kapil@cs.columbia.edu

Abstract

Sentence compression has been shown to benefit from joint inference involving both n-gram and dependency-factored objectives but this typically requires expensive integer programming. We explore instead the use of Lagrangian relaxation to decouple the two subproblems and solve them separately. While dynamic programming is viable for bigram-based sentence compression, finding optimal compressed trees within graphs is NP-hard. We recover approximate solutions to this problem using LP relaxation and maximum spanning tree algorithms, yielding techniques that can be combined with the efficient bigram-based inference approach using Lagrange multipliers. Experiments show that these approximation strategies produce results comparable to a state-of-the-art integer linear programming formulation for the same joint inference task along with a significant improvement in runtime.

1 Introduction

Sentence compression is a text-to-text generation task in which an input sentence must be transformed into a shorter output sentence which accurately reflects the meaning in the input and also remains grammatically well-formed. The compression task has received increasing attention in recent years, in part due to the availability of datasets such as the Ziff-Davis corpus (Knight and Marcu, 2000) and the Edinburgh compression corpora (Clarke and Lapata, 2006), from which the following example is drawn.

Original: In 1967 Chapman, who had cultivated a conventional image with his ubiquitous tweed jacket and pipe, by his own later admission stunned a party attended by his friends and future Python colleagues by coming out as a homosexual.

Compressed: In 1967 Chapman, who had cultivated a conventional image, stunned a party by coming out as a homosexual.

Following an assumption often used in compression systems, the compressed output in this corpus is constructed by dropping tokens from the input sentence without any paraphrasing or reordering.[1]

A number of diverse approaches have been proposed for deletion-based sentence compression, including techniques that assemble the output text under an n-gram factorization over the input text (McDonald, 2006; Clarke and Lapata, 2008) or an arc factorization over input dependency parses (Filippova and Strube, 2008; Galanis and Androutsopoulos, 2010; Filippova and Altun, 2013). Joint methods have also been proposed that invoke integer linear programming (ILP) formulations to simultaneously consider multiple structural inference problems—both over n-grams and input dependencies (Martins and Smith, 2009) or n-grams and all possible dependencies (Thadani and McKeown, 2013). However, it is well-established that the utility of ILP for optimal inference in structured problems is often outweighed by the worst-case performance of ILP solvers on large problems without unique integral solutions. Furthermore, approximate solutions can often be adequate for real-world generation systems, particularly in the presence of linguistically-motivated constraints such as those described by Clarke and Lapata (2008), or domain-specific

[1] This is referred to as *extractive compression* by Cohn and Lapata (2008) & Galanis and Androutsopoulos (2010) following the terminology used in document summarization.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1241–1251,
Baltimore, Maryland, USA, June 23-25 2014.

pruning strategies such as the use of sentence templates to constrain the output.

In this work, we develop approximate inference strategies to the joint approach of Thadani and McKeown (2013) which trade the optimality guarantees of exact ILP for faster inference by separately solving the n-gram and dependency subproblems and using Lagrange multipliers to enforce consistency between their solutions. However, while the former problem can be solved efficiently using the dynamic programming approach of McDonald (2006), there are no efficient algorithms to recover maximum weighted non-projective subtrees in a general directed graph. Maximum spanning tree algorithms, commonly used in non-projective dependency parsing (McDonald et al., 2005), are not easily adaptable to this task since the maximum-weight subtree is not necessarily a part of the maximum spanning tree.

We therefore consider methods to recover approximate solutions for the subproblem of finding the maximum weighted subtree in a graph, common among which is the use of a linear programming relaxation. This linear program (LP) appears empirically tight for compression problems and our experiments indicate that simply using the non-integral solutions of this LP in Lagrangian relaxation can empirically lead to reasonable compressions. In addition, we can recover approximate solutions to this problem by using the Chu-Liu Edmonds algorithm for recovering maximum spanning trees (Chu and Liu, 1965; Edmonds, 1967) over the relatively sparse subgraph defined by a solution to the relaxed LP. Our proposed approximation strategies are evaluated using automated metrics in order to address the question: under what conditions should a real-world sentence compression system implementation consider exact inference with an ILP or approximate inference? The contributions of this work include:

- An empirically-useful technique for approximating the maximum-weight subtree in a weighted graph using LP-relaxed inference.
- Multiple approaches to generate good approximate solutions for joint multi-structure compression, based on Lagrangian relaxation to enforce equality between the sequential and syntactic inference subproblems.
- An analysis of the tradeoffs incurred by joint approaches with regard to runtime as well as performance under automated measures.

2 Multi-Structure Sentence Compression

Even though compression is typically formulated as a token deletion task, it is evident that dropping tokens independently from an input sentence will likely not result in fluent and meaningful compressive text. Tokens in well-formed sentences participate in a number of syntactic and semantic relationships with other tokens, so one might expect that accounting for heterogenous structural relationships between tokens will improve the coherence of the output sentence. Furthermore, much recent work has focused on the challenge of joint sentence extraction and compression, also known as *compressive summarization* (Martins and Smith, 2009; Berg-Kirkpatrick et al., 2011; Almeida and Martins, 2013; Li et al., 2013; Qian and Liu, 2013), in which questions of efficiency are paramount due to the larger problems involved; however, these approaches largely restrict compression to pruning parse trees, thereby imposing a dependency on parser performance. We focus in this work on a sentence-level compression system to approximate the ILP-based inference of Thadani and McKeown (2013) which does not restrict compressions to follow input parses but permits the generation of novel dependency relations in output compressions.

The rest of this section is organized as follows: §2.1 provies an overview of the joint sequential and syntactic objective for compression from Thadani and McKeown (2013) while §2.2 discusses the use of Lagrange multipliers to enforce consistency between the different structures considered. Following this, §2.3 discusses a dynamic program to find maximum weight bigram subsequences from the input sentence, while §2.4 covers LP relaxation-based approaches for approximating solutions to the problem of finding a maximum-weight subtree in a graph of potential output dependencies. Finally, §2.5 discusses the features and model training approach used in our experimental results which are presented in §3.

2.1 Joint objective

We begin with some notation. For an input sentence S comprised of n tokens including duplicates, we denote the set of tokens in S by $T \triangleq \{t_i : 1 \leq i \leq n\}$. Let C represent a compression of S and let $x_i \in \{0, 1\}$ denote an indicator variable whose value corresponds to whether token $t_i \in T$ is present in the compressed sentence

C. In addition, we define bigram indicator variables $y_{ij} \in \{0, 1\}$ to represent whether a particular order-preserving bigram[2] $\langle t_i, t_j \rangle$ from S is present as a contiguous bigram in C as well as dependency indicator variables $z_{ij} \in \{0, 1\}$ corresponding to whether the dependency arc $t_i \rightarrow t_j$ is present in the dependency parse of C. The score for a given compression C can now be defined to factor over its tokens, n-grams and dependencies as follows.

$$\begin{aligned} score(C) \;=\; & \sum_{t_i \in T} x_i \cdot \theta_{\text{tok}}(t_i) \\ & + \sum_{\substack{t_i \in T \cup \{\text{START}\}, \\ t_j \in T \cup \{\text{END}\}}} y_{ij} \cdot \theta_{\text{bgr}}(\langle t_i, t_j \rangle) \\ & + \sum_{\substack{t_i \in T \cup \{\text{ROOT}\}, \\ t_j \in T}} z_{ij} \cdot \theta_{\text{dep}}(t_i \rightarrow t_j) \end{aligned} \quad (1)$$

where θ_{tok}, θ_{bgr} and θ_{dep} are feature-based scoring functions for tokens, bigrams and dependencies respectively. Specifically, each $\theta_v(\cdot) \equiv \mathbf{w}_v^\top \boldsymbol{\phi}_v(\cdot)$ where $\boldsymbol{\phi}_v(\cdot)$ is a feature map for a given variable type $v \in \{\text{tok, bgr, dep}\}$ and $\mathbf{w}_v$ is the corresponding vector of learned parameters.

The inference task involves recovering the highest scoring compression C^* under a particular set of model parameters $\mathbf{w}$.

$$\begin{aligned} C^* &= \arg\max_C \; score(C) \\ &= \arg\max_{\mathbf{x},\mathbf{y},\mathbf{z}} \; \mathbf{x}^\top \boldsymbol{\theta}_{\text{tok}} + \mathbf{y}^\top \boldsymbol{\theta}_{\text{bgr}} + \mathbf{z}^\top \boldsymbol{\theta}_{\text{dep}} \end{aligned} \quad (2)$$

where the incidence vector $\mathbf{x} \triangleq \langle x_i \rangle_{t_i \in T}$ represents an entire token configuration over T, with $\mathbf{y}$ and $\mathbf{z}$ defined analogously to represent configurations of bigrams and dependencies. $\boldsymbol{\theta}_v \triangleq \langle \theta_v(\cdot) \rangle$ denotes a corresponding vector of scores for each variable type v under the current model parameters. In order to recover meaningful compressions by optimizing (2), the inference step must ensure:

1. The configurations $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ are *consistent* with each other, i.e., all configurations cover the same tokens.
2. The structural configurations $\mathbf{y}$ and $\mathbf{z}$ are *non-degenerate*, i.e, the bigram configuration $\mathbf{y}$ represents an acyclic path while the dependency configuration $\mathbf{z}$ forms a tree.

These requirements naturally rule out simple approximate inference formulations such as search-based approaches for the joint objective.[3] An ILP-based inference solution is demonstrated in Thadani and McKeown (2013) that makes use of linear constraints over the boolean variables x_i, y_{ij} and z_{ij} to guarantee consistency, as well as auxiliary real-valued variables and constraints representing the flow of commodities (Magnanti and Wolsey, 1994) in order to establish structure in $\mathbf{y}$ and $\mathbf{z}$. In the following section, we propose an alternative formulation that exploits the modularity of this joint objective.

2.2 Lagrangian relaxation

Dual decomposition (Komodakis et al., 2007) and Lagrangian relaxation in general are often used for solving joint inference problems which are decomposable into individual subproblems linked by equality constraints (Koo et al., 2010; Rush et al., 2010; Rush and Collins, 2011; DeNero and Macherey, 2011; Martins et al., 2011; Das et al., 2012; Almeida and Martins, 2013). This approach permits sub-problems to be solved separately using problem-specific efficient algorithms, while consistency over the structures produced is enforced through Lagrange multipliers via iterative optimization. Exact solutions are guaranteed when the algorithm converges on a consistent primal solution, although this convergence itself is not guaranteed and depends on the tightness of the underlying LP relaxation. The primary advantage of this technique is the ability to leverage the underlying structure of the problems in inference rather than relying on a generic ILP formulation while still often producing exact solutions.

The multi-structure inference problem described in the previous section seems in many ways to be a natural fit to such an approach since output scores factor over different types of structure that comprise the output compression. Even if ILP-based approaches perform reasonably at the scale of single-sentence compression problems, the exponential worst-case complexity of general-purpose ILPs will inevitably pose challenges when scaling up to (a) handle larger inputs, (b) use higher-order structural fragments, or (c) incorporate additional models.

[2] Although Thadani and McKeown (2013) is not restricted to bigrams or order-preserving n-grams, we limit our discussion to this scenario as it also fits the assumptions of McDonald (2006) and the datasets of Clarke and Lapata (2006).

[3] This work follows Thadani and McKeown (2013) in recovering non-projective trees for inference. However, recovering projective trees is tractable when a total ordering of output tokens is assumed. This will be addressed in future work.

Consider once more the optimization problem characterized by (2) The two structural problems that need to be solved in this formulation are the extraction of a maximum-weight acyclic subsequence of bigrams $\mathbf{y}$ from the lattice of all order-preserving bigrams from S and the recovery of a maximum-weight directed subtree $\mathbf{z}$. Let $\boldsymbol{\alpha}(\mathbf{y}) \in \{0,1\}^n$ denote the incidence vector of tokens contained in the n-gram sequence $\mathbf{y}$ and $\boldsymbol{\beta}(\mathbf{z}) \in \{0,1\}^n$ denote the incidence vector of words contained in the dependency tree $\mathbf{z}$. We can now rewrite the objective in (2) while enforcing the constraint that the words contained in the sequence $\mathbf{y}$ are the same as the words contained in the tree $\mathbf{z}$, i.e., $\boldsymbol{\alpha}(\mathbf{y}) = \boldsymbol{\beta}(\mathbf{z})$, by introducing a vector of Lagrange multipliers $\boldsymbol{\lambda} \in \mathbb{R}^n$. In addition, the token configuration $\mathbf{x}$ can be rewritten in the form of a weighted combination of $\boldsymbol{\alpha}(\mathbf{y})$ and $\boldsymbol{\beta}(\mathbf{z})$ to ensure its consistency with $\mathbf{y}$ and $\mathbf{z}$. This results in the following Lagrangian:

$$\begin{aligned} L(\boldsymbol{\lambda}, \mathbf{y}, \mathbf{z}) = \mathbf{y}^\top \boldsymbol{\theta}_{\text{bgr}} + \mathbf{z}^\top \boldsymbol{\theta}_{\text{dep}} \\ + \boldsymbol{\theta}_{\text{tok}}^\top (\psi \cdot \boldsymbol{\alpha}(\mathbf{y}) + (1-\psi) \cdot \boldsymbol{\beta}(\mathbf{z})) \\ + \boldsymbol{\lambda}^\top (\boldsymbol{\alpha}(\mathbf{y}) - \boldsymbol{\beta}(\mathbf{z})) \end{aligned} \tag{3}$$

Finding the $\mathbf{y}$ and $\mathbf{z}$ that maximize this Lagrangian above yields a dual objective, and the dual problem corresponding to the primal objective specified in (2) is therefore the minimization of this objective over the Lagrange multipliers $\boldsymbol{\lambda}$.

$$\begin{aligned} &\min_{\boldsymbol{\lambda}} \max_{\mathbf{y},\mathbf{z}} L(\boldsymbol{\lambda}, \mathbf{y}, \mathbf{z}) \\ =&\min_{\boldsymbol{\lambda}} \max_{\mathbf{y}} \mathbf{y}^\top \boldsymbol{\theta}_{\text{bgr}} + (\boldsymbol{\lambda} + \psi \cdot \boldsymbol{\theta}_{\text{tok}})^\top \boldsymbol{\alpha}(\mathbf{y}) \\ &+ \max_{\mathbf{z}} \mathbf{z}^\top \boldsymbol{\theta}_{\text{dep}} - (\boldsymbol{\lambda} + (\psi - 1) \cdot \boldsymbol{\theta}_{\text{tok}})^\top \boldsymbol{\beta}(\mathbf{z}) \\ =&\min_{\boldsymbol{\lambda}} \max_{\mathbf{y}} f(\mathbf{y}, \boldsymbol{\lambda}, \psi, \boldsymbol{\theta}) \\ &+ \max_{\mathbf{z}} g(\mathbf{z}, \boldsymbol{\lambda}, \psi, \boldsymbol{\theta}) \end{aligned} \tag{4}$$

This can now be solved with the iterative subgradient algorithm illustrated in Algorithm 1. In each iteration i, the algorithm solves for $\mathbf{y}^{(i)}$ and $\mathbf{z}^{(i)}$ under $\boldsymbol{\lambda}^{(i)}$, then generates $\boldsymbol{\lambda}^{(i+1)}$ to penalize inconsistencies between $\boldsymbol{\alpha}(\mathbf{y}^{(i)})$ and $\boldsymbol{\beta}(\mathbf{z}^{(i)})$. When $\boldsymbol{\alpha}(\mathbf{y}^{(i)}) = \boldsymbol{\beta}(\mathbf{z}^{(i)})$, the resulting primal solution is exact, i.e., $\mathbf{y}^{(i)}$ and $\mathbf{z}^{(i)}$ represent the optimal structures under (2). Otherwise, if the algorithm starts oscillating between a few primal solutions, the underlying LP must have a non-integral solution in which case approximation heuristics can be employed.[4] The application of this Lagrangian relaxation strategy is contingent upon the existence of algorithms to solve the maximization subproblems for $f(\mathbf{y}, \boldsymbol{\lambda}, \psi, \boldsymbol{\theta})$ and $g(\mathbf{z}, \boldsymbol{\lambda}, \psi, \boldsymbol{\theta})$. The following sections discuss our approach to these problems.

Algorithm 1 Subgradient-based joint inference

Input: scores $\boldsymbol{\theta}$, ratio ψ, repetition limit l_{max}, iteration limit i_{max}, learning rate schedule $\boldsymbol{\eta}$
Output: token configuration $\mathbf{x}$

$\boldsymbol{\lambda}^{(0)} \leftarrow \langle 0 \rangle^n$
$M \leftarrow \varnothing, M_{\text{repeats}} \leftarrow \varnothing$
for iteration $i < i_{\text{max}}$ **do**
  $\hat{\mathbf{y}} \leftarrow \arg\max_{\mathbf{y}} f(\mathbf{y}, \boldsymbol{\lambda}, \psi, \boldsymbol{\theta})$
  $\hat{\mathbf{z}} \leftarrow \arg\max_{\mathbf{z}} g(\mathbf{z}, \boldsymbol{\lambda}, \psi, \boldsymbol{\theta})$
  if $\boldsymbol{\alpha}(\hat{\mathbf{y}}) = \boldsymbol{\beta}(\hat{\mathbf{z}})$ **then return** $\boldsymbol{\alpha}(\hat{\mathbf{y}})$
  if $\boldsymbol{\alpha}(\hat{\mathbf{y}}) \in M$ **then**
    $M_{\text{repeats}} \leftarrow M_{\text{repeats}} \cup \{\boldsymbol{\alpha}(\hat{\mathbf{y}})\}$
  if $\boldsymbol{\beta}(\hat{\mathbf{z}}) \in M$ **then**
    $M_{\text{repeats}} \leftarrow M_{\text{repeats}} \cup \{\boldsymbol{\beta}(\hat{\mathbf{z}})\}$
  if $|M_{\text{repeats}}| \geq l_{\text{max}}$ **then break**
  $M \leftarrow M \cup \{\boldsymbol{\alpha}(\hat{\mathbf{y}}), \boldsymbol{\beta}(\hat{\mathbf{z}})\}$
  $\boldsymbol{\lambda}^{(i+1)} \leftarrow \boldsymbol{\lambda}^{(i)} - \eta_i \left(\boldsymbol{\alpha}(\hat{\mathbf{y}}) - \boldsymbol{\beta}(\hat{\mathbf{z}})\right)$
return $\arg\max_{\mathbf{x} \in \mathbf{M}_{\text{repeats}}} score(\mathbf{x})$

2.3 Bigram subsequences

McDonald (2006) provides a Viterbi-like dynamic programming algorithm to recover the highest-scoring sequence of order-preserving bigrams from a lattice, either in unconstrained form or with a specific length constraint. The latter requires a dynamic programming table $Q[i][r]$ which represents the best score for a compression of length r ending at token i. The table can be populated using the following recurrence:

$$\begin{aligned} Q[i][1] &= score(S, \text{START}, i) \\ Q[i][r] &= \max_{j<i} Q[j][r-1] + score(S, i, j) \\ Q[i][R+1] &= Q[i][R] + score(S, i, \text{END}) \end{aligned}$$

where R is the required number of output tokens and the scoring function is defined as

$$score(S, i, j) \triangleq \theta_{\text{bgr}}(\langle t_i, t_j \rangle) + \lambda_j + \psi \cdot \theta_{\text{tok}}(t_j)$$

so as to solve $f(\mathbf{y}, \boldsymbol{\lambda}, \psi, \boldsymbol{\theta})$ from (4). This approach requires $O(n^2 R)$ time in order to identify

[4] Heuristic approaches (Komodakis et al., 2007; Rush et al., 2010), tightening (Rush and Collins, 2011) or branch and bound (Das et al., 2012) can still be used to retrieve optimal solutions, but we did not explore these strategies here.

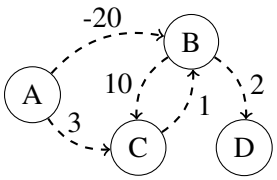


Figure 1: An example of the difficulty of recovering the maximum-weight subtree (B→C, B→D) from the maximum spanning tree (A→C, C→B, B→D).

the highest scoring sequence $\mathbf{y}$ and corresponding token configuration $\boldsymbol{\alpha}(\mathbf{y})$.

2.4 Dependency subtrees

The maximum-weight non-projective subtree problem over general graphs is not as easily solved. Although the maximum *spanning* tree for a given token configuration can be recovered efficiently, Figure 1 illustrates that the maximum-scoring subtree is not necessarily found within it. The problem of recovering a maximum-weight subtree in a graph has been shown to be NP-hard even with uniform edge weights (Lau et al., 2006).

In order to produce a solution to this subproblem, we use an LP relaxation of the relevant portion of the ILP from Thadani and McKeown (2013) by omitting integer constraints over the token and dependency variables in $\mathbf{x}$ and $\mathbf{z}$ respectively. For simplicity, however, we describe the ILP version rather than the relaxed LP in order to motivate the constraints with their intended purpose rather than their effect in the relaxed problem. The objective for this LP is given by

$$\max_{\mathbf{x},\mathbf{z}} \; \mathbf{x}^\top \boldsymbol{\theta'}_{\text{tok}} + \mathbf{z}^\top \boldsymbol{\theta}_{\text{dep}} \tag{5}$$

where the vector of token scores is redefined as

$$\boldsymbol{\theta'}_{\text{tok}} \triangleq (1 - \psi) \cdot \boldsymbol{\theta}_{\text{tok}} - \boldsymbol{\lambda} \tag{6}$$

in order to solve $g(\mathbf{z}, \boldsymbol{\lambda}, \psi, \boldsymbol{\theta})$ from (4).

Linear constraints are introduced to produce dependency structures that are close to the optimal dependency trees. First, tokens in the solution must only be active if they have a single active incoming dependency edge. In addition, to avoid producing multiple disconnected subtrees, only one dependency is permitted to attach to the ROOT pseudo-token.

$$x_j - \sum_i z_{ij} = 0, \quad \forall t_j \in T \tag{7}$$

$$\sum_j z_{ij} = 1, \quad \text{if } t_i = \text{ROOT} \tag{8}$$

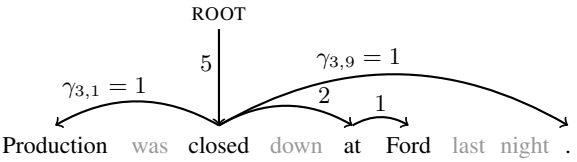


Figure 2: An illustration of commodity values for a valid solution of the non-relaxed ILP.

In order to avoid cycles in the dependency tree, we include additional variables to establish *single-commodity flow* (Magnanti and Wolsey, 1994) between all pairs of tokens. These γ_{ij} variables carry non-negative real values which must be consumed by active tokens that they are incident to.

$$\gamma_{ij} \geq 0, \quad \forall t_i, t_j \in T \tag{9}$$

$$\sum_i \gamma_{ij} - \sum_k \gamma_{jk} = x_j, \quad \forall t_j \in T \tag{10}$$

These constraints ensure that cyclic structures are not possible in the non-relaxed ILP. In addition, they serve to establish connectivity for the dependency structure $\mathbf{z}$ since commodity can only originate in one location—at the pseudo-token ROOT which has no incoming commodity variables. However, in order to enforce these properties on the output dependency structure, this acyclic, connected commodity structure must constrain the activation of the z variables.

$$\gamma_{ij} - C_{\text{max}} z_{ij} \leq 0, \quad \forall t_i, t_j \in T \tag{11}$$

where C_{max} is an arbitrary upper bound on the value of γ_{ij} variables. Figure 2 illustrates how these commodity flow variables constrain the output of the ILP to be a tree. However, the effect of these constraints is diminished when solving an LP relaxation of the above problem.

In the LP relaxation, x_i and z_{ij} are redefined as real-valued variables in $[0, 1]$, potentially resulting in fractional values for dependency and token indicators. As a result, the commodity flow network is able to establish connectivity but cannot enforce a tree structure, for instance, directed acyclic structures are possible and token indicators x_i may be partially be assigned to the solution structure. This poses a challenge in implementing $\boldsymbol{\beta}(\mathbf{z})$ which is needed to recover a token configuration from the solution of this subproblem.

We propose two alternative solutions to address this issue in the context of the joint inference strategy. The first is to simply use the relaxed token configuration identified by the LP in Algorithm 1,

i.e., to set $\boldsymbol{\beta}(\tilde{\mathbf{z}}) = \tilde{x}$ where $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{z}}$ represent the real-valued counterparts of the incidence vectors $\mathbf{x}$ and $\mathbf{z}$. The viability of this approximation strategy is due to the following:

- The relaxed LP is empirically fairly tight, yielding integral solutions 89% of the time on the compression datasets described in §3.
- The bigram subproblem is guaranteed to return a well-formed integral solution which obeys the imposed compression rate, so we are assured of a source of valid—if non-optimal—solutions in line 13 of Algorithm 1.

We also consider another strategy that attempts to approximate a valid integral solution to the dependency subproblem. In order to do this, we first include an additional constraint in the relaxed LP which restrict the number of tokens in the output to a specific number of tokens R that is given by an input compression rate.

$$\sum_i x_i = R \tag{12}$$

The addition of this constraint to the relaxed LP reduces the rate of integral solutions drastically—from 89% to approximately 33%—but it serves to ensure that the resulting token configuration $\tilde{\mathbf{x}}$ has at least as many non-zero elements as R, i.e., there are at least as many tokens activated in the LP solution as are required in a valid solution.

We then construct a subgraph $G(\tilde{\mathbf{z}})$ consisting of all dependency edges that were assigned non-zero values in the solution, assigning to each edge a score equal to the score of that edge in the LP as well as the score of its dependent word, i.e., each z_{ij} in $G(\tilde{\mathbf{z}})$ is assigned a score of $\theta_{\text{dep}}(\langle t_i, t_j \rangle) - \lambda_j + (1 - \psi) \cdot \theta_{\text{tok}}(t_j)$. Since the commodity flow constraints in (9)–(11) ensure a connected $\tilde{\mathbf{z}}$, it is therefore possible to recover a maximum-weight spanning tree from $G(\tilde{\mathbf{z}})$ using the Chu-Liu Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967).[5] Although the runtime of this algorithm is cubic in the size of the input graph, it is fairly speedy when applied on relatively sparse graphs such as the solutions to the LP described above. The resulting spanning tree is a useful integral approximation of $\tilde{\mathbf{z}}$ but, as mentioned previously, may contain more nodes than R due to fractional values in $\tilde{\mathbf{x}}$; we therefore repeatedly prune leaves with the lowest incoming edge weight in the current tree until exactly R nodes remain. The resulting tree is assumed to be a reasonable approximation of the optimal integral solution to this LP.

The Chu-Liu Edmonds algorithm is also employed for another purpose: when the underlying LP for the joint inference problem is not tight—a frequent occurrence in our compression experiments—Algorithm 1 will not converge on a single primal solution and will instead oscillate between solutions that are close to the dual optimum. We identify this phenomenon by counting repeated solutions and, if they exceed some threshold l_{max} with at least one repeated solution from either subproblem, we terminate the update procedure for Lagrange multipliers and instead attempt to identify a good solution from the repeating ones by scoring them under (2). It is straightforward to recover and score a bigram configuration $\mathbf{y}$ from a token configuration $\boldsymbol{\beta}(\mathbf{z})$. However, scoring solutions produced by the dynamic program from §2.3 also requires the score over a corresponding parse tree; this can be recovered by constructing a dependency subgraph containing across only the tokens that are active in $\boldsymbol{\alpha}(\mathbf{y})$ and retrieving the maximum spanning tree for that subgraph using the Chu-Liu Edmonds algorithm.

2.5 Learning and Features

The features used in this work are largely based on the features from Thadani and McKeown (2013).

- $\boldsymbol{\phi}_{\text{tok}}$ contains features for part-of-speech (POS) tag sequences of length up to 3 around the token, features for the dependency label of the token conjoined with its POS, lexical features for verb stems and non-word symbols and morphological features that identify capitalized sequences, negations and words in parentheses.
- $\boldsymbol{\phi}_{\text{bgr}}$ contains features for POS patterns in a bigram, the labels of dependency edges incident to it, its likelihood under a Gigaword language model (LM) and an indicator for whether it is present in the input sentence.
- $\boldsymbol{\phi}_{\text{dep}}$ contains features for the probability of a dependency edge under a smoothed dependency grammar constructed from the Penn Treebank and various conjunctions of the following features: (a) whether the edge appears as a dependency or ancestral relation in the input parse (b) the directionality of the depen-

[5] A detailed description of the Chu-Liu Edmonds algorithm for MSTs is available in McDonald et al. (2005).

dency (c) the label of the edge (d) the POS tags of the tokens incident to the edge and (e) the labels of their surrounding chunks and whether the edge remains within the chunk.

For the experiments in the following section, we trained models using a variant of the structured perceptron (Collins, 2002) which incorporates minibatches (Zhao and Huang, 2013) for easy parallelization and faster convergence.[6] Overfitting was avoided by averaging parameters and monitoring performance against a held-out development set during training. All models were trained using variants of the ILP-based inference approach of Thadani and McKeown (2013). We followed Martins et al. (2009) in using LP-relaxed inference during learning, assuming algorithmic separability (Kulesza and Pereira, 2007) for these problems.

3 Experiments

We ran compression experiments over the newswire (NW) and broadcast news transcription (BN) corpora compiled by Clarke and Lapata (2008) which contain gold compressions produced by human annotators using only word deletion. The datasets were filtered to eliminate instances with less than 2 and more than 110 tokens for parser compatibility and divided into training/development/test sections following the splits from Clarke and Lapata (2008), yielding 953/63/603 instances for the NW corpus and 880/78/404 for the BN corpus. Gold dependency parses were approximated by running the Stanford dependency parser[7] over reference compressions.

Following evaluations in machine translation as well as previous work in sentence compression (Unno et al., 2006; Clarke and Lapata, 2008; Martins and Smith, 2009; Napoles et al., 2011b; Thadani and McKeown, 2013), we evaluate system performance using F_1 metrics over n-grams and dependency edges produced by parsing system output with RASP (Briscoe et al., 2006) and the Stanford parser. All ILPs and LPs were solved using Gurobi,[8] a high-performance commercial-grade solver. Following a recent analysis of compression evaluations (Napoles et al., 2011b) which revealed a strong correlation between system compression rate and human judgments of compression quality, we constrained all systems to produce compressed output at a specific rate—determined by the the gold compressions available for each instance—to ensure that the reported differences between the systems under study are meaningful.

3.1 Systems

We report results over the following systems grouped into three categories of models: tokens + n-grams, tokens + dependencies, and joint models.

- **3-LM:** A reimplementation of the unsupervised ILP of Clarke and Lapata (2008) which infers order-preserving trigram variables parameterized with log-likelihood under an LM and a significance score for token variables inspired by Hori and Furui (2004), as well as various linguistically-motivated constraints to encourage fluency in output compressions.
- **DP:** The bigram-based dynamic program of McDonald (2006) described in §2.3.[9]
- **LP→MST:** An approximate inference approach based on an LP relaxation of **ILP-Dep**. As discussed in §2.4, a maximum spanning tree is recovered from the output of the LP and greedily pruned in order to generate a valid integral solution while observing the imposed compression rate.
- **ILP-Dep:** A version of the joint ILP of Thadani and McKeown (2013) without n-gram variables and corresponding features.
- **DP+LP→MST:** An approximate joint inference approach based on Lagrangian relaxation that uses **DP** for the maximum weight subsequence problem and **LP→MST** for the maximum weight subtree problem.
- **DP+LP:** Another Lagrangian relaxation approach that pairs **DP** with the non-integral solutions from an LP relaxation of the maximum weight subtree problem (cf. §2.4).
- **ILP-Joint:** The full ILP from Thadani and McKeown (2013), which provides an upper bound on the performance of the proposed approximation strategies.

The learning rate schedule for the Lagrangian relaxation approaches was set as $\eta_i \triangleq \tau/(\tau + i)$,[10] while the hyperparameter ψ was tuned using the

[6] We used a minibatch size of 4 in all experiments.

[7] `http://nlp.stanford.edu/software/`

[8] `http://www.gurobi.com`

[9] For consistent comparisons with the other systems, our reimplementation does not include the k-best inference strategy presented in McDonald (2006) for learning with MIRA.

[10] τ was set to 100 for aggressive subgradient updates.

Inference objective	Inference technique	n-grams $F_1\%$ $n=1$	2	3	4	Syntactic relations $F_1\%$ z	Stanford	RASP	Inference time (s)
n-grams	3-LM (CL08)	74.96	60.60	46.83	38.71	-	60.52	57.49	0.72
	DP (McD06)	78.80	66.04	52.67	42.39	-	63.28	57.89	0.01
deps	LP→MST	**79.61**	64.32	50.36	40.97	66.57	66.82	59.70	0.07
	ILP-Dep	**80.02**	65.99	52.42	43.07	**72.43**	67.63	60.78	0.16
joint	DP + LP→MST	79.50	66.75	53.48	44.33	64.63	67.69	60.94	0.24
	DP + LP	79.10	**68.22**	**55.05**	**45.81**	65.74	**68.24**	**62.04**	0.12
	ILP-Joint (TM13)	**80.13**	**68.34**	**55.56**	**46.60**	**72.57**	**68.89**	**62.61**	0.31

Table 1: Experimental results for the BN corpus, averaged over 3 gold compressions per instance. All systems were restricted to compress to the size of the median gold compression yielding an average compression rate of 77.26%.

Inference objective	Inference technique	n-grams $F_1\%$ $n=1$	2	3	4	Syntactic relations $F_1\%$ z	Stanford	RASP	Inference time (s)
n-grams	3-LM (CL08)	66.66	51.59	39.33	30.55	-	50.76	49.57	1.22
	DP (McD06)	73.18	58.31	45.07	34.77	-	56.23	51.14	0.01
deps	LP→MST	73.32	55.12	41.18	31.44	61.01	58.37	52.57	0.12
	ILP-Dep	**73.76**	57.09	43.47	33.44	**65.45**	60.06	54.31	0.28
joint	DP + LP→MST	73.13	57.03	43.79	34.01	57.91	58.46	53.20	0.33
	DP + LP	72.06	**59.83**	**47.39**	**37.72**	58.13	58.97	53.78	0.21
	ILP-Joint (TM13)	**74.00**	**59.90**	**47.22**	**37.01**	**65.65**	**61.29**	**56.24**	0.60

Table 2: Experimental results for the NW corpus with all systems compressing to the size of the gold compression, yielding an average compression rate of 70.24%. In both tables, bold entries show significant gains within a column under the paired t-test ($p < 0.05$) and Wilcoxon's signed rank test ($p < 0.01$).

development split of each corpus.[11]

3.2 Results

Tables 1 and 2 summarize the results from our compression experiments on the BN and NW corpora respectively. Starting with the n-gram approaches, the performance of **3-LM** leads us to observe that the gains of supervised learning far outweigh the utility of higher-order n-gram factorization, which is also responsible for a significant increase in wall-clock time. In contrast, **DP** is an order of magnitude faster than all other approaches studied here although it is not competitive under parse-based measures such as RASP $F_1\%$ which is known to correlate with human judgments of grammaticality (Clarke and Lapata, 2006).

We were surprised by the strong performance of the dependency-based inference techniques, which yielded results that approached the joint model in both n-gram and parse-based measures. The exact **ILP-Dep** approach halves the runtime of **ILP-Joint** to produce compressions that have similar (although statistically distinguishable) scores. Approximating dependency-based inference with **LP→MST** yields similar performance for a further halving of runtime; however, the performance of this approach is notably worse.

Turning to the joint approaches, the strong performance of **ILP-Joint** is expected; less so is the relatively high but yet practically reasonable runtime that it requires. We note, however, that these ILPs are solved using a highly-optimized commercial-grade solver that can utilize all CPU cores[12] while our approximation approaches are implemented as single-processed Python code without significant effort toward optimization. Comparing the two approximation strategies shows a clear performance advantage for **DP+LP** over **DP+LP→MST**: the latter approach entails slower inference due to the overhead of running the Chu-Liu Edmonds algorithm at every dual update, and furthermore, the error introduced by approximating an integral solution re-

[11] We were surprised to observe that performance improved significantly when ψ was set closer to 1, thereby emphasizing token features in the dependency subproblem. The final values chosen were $\psi_{\text{BN}} = 0.9$ and $\psi_{\text{NW}} = 0.8$.

[12] 16 cores in our experimental environment.

sults in a significant decrease in dependency recall. In contrast, **DP+LP** directly optimizes the dual problem by using the relaxed dependency solution to update Lagrange multipliers and achieves the best performance on parse-based F_1 outside of the slower ILP approaches. Convergence rates also vary for these two techniques: **DP+LP** has a lower rate of empirical convergence (15% on BN and 4% on NW) when compared to **DP+LP→MST** (19% on BN and 6% on NW).

Figure 3 shows the effect of input sentence length on inference time and performance for **ILP-Joint** and **DP+LP** over the NW test corpus.[13] The timing results reveal that the approximation strategy is consistently faster than the ILP solver. The variation in RASP F_1% with input size indicates the viability of a hybrid approach which could balance accuracy and speed by using **ILP-Joint** for smaller problems and **DP+LP** for larger ones.

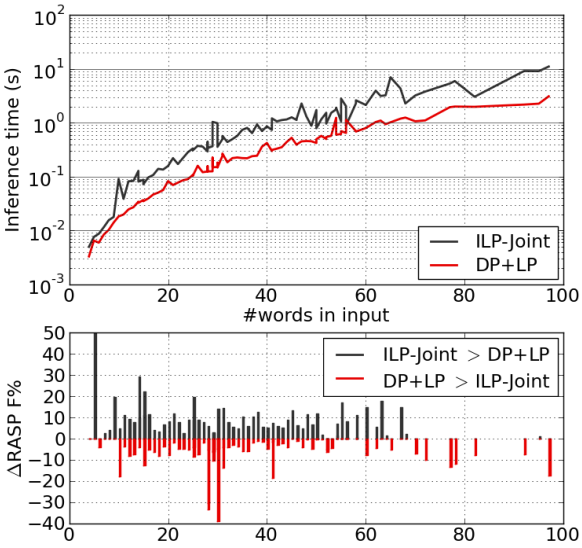


Figure 3: Effect of input size on (a) inference time, and (b) the corresponding difference in RASP F_1% (**ILP-Joint** – **DP+LP**) on the NW corpus.

4 Related Work

Sentence compression is one of the better-studied text-to-text generation problems and has been observed to play a significant role in human summarization (Jing, 2000; Jing and McKeown, 2000). Most approaches to sentence compression are supervised (Knight and Marcu, 2002; Riezler et al., 2003; Turner and Charniak, 2005; McDonald, 2006; Unno et al., 2006; Galley and McKeown, 2007; Nomoto, 2007; Cohn and Lapata, 2009; Galanis and Androutsopoulos, 2010; Ganitkevitch et al., 2011; Napoles et al., 2011a; Filippova and Altun, 2013) following the release of datasets such as the Ziff-Davis corpus (Knight and Marcu, 2000) and the Edinburgh compression corpora (Clarke and Lapata, 2006; Clarke and Lapata, 2008), although unsupervised approaches—largely based on ILPs—have also received consideration (Clarke and Lapata, 2007; Clarke and Lapata, 2008; Filippova and Strube, 2008). Compression has also been used as a tool for document summarization (Daumé and Marcu, 2002; Zajic et al., 2007; Clarke and Lapata, 2007; Martins and Smith, 2009; Berg-Kirkpatrick et al., 2011; Woodsend and Lapata, 2012; Almeida and Martins, 2013; Molina et al., 2013; Li et al., 2013; Qian and Liu, 2013), with recent work formulating the summarization task as joint sentence extraction and compression and often employing ILP or Lagrangian relaxation. Monolingual compression also faces many obstacles common to decoding in machine translation, and a number of approaches which have been proposed to combine phrasal and syntactic models (Huang and Chiang, 2007; Rush and Collins, 2011) *inter alia* offer directions for future research into compression problems.

5 Conclusion

We have presented approximate inference strategies to jointly compress sentences under bigram and dependency-factored objectives by exploiting the modularity of the task and considering the two subproblems in isolation. Experiments show that one of these approximation strategies produces results comparable to a state-of-the-art integer linear program for the same joint inference task with a 60% reduction in average inference time.

Acknowledgments

The author is grateful to Alexander Rush for helpful discussions and to the anonymous reviewers for their comments. This work was supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center (DoI/NBC) contract number D11PC20153. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.[14]

[13] Similar results were observed for the BN test corpus.

[14] The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

References

Miguel Almeida and André F. T. Martins. 2013. Fast and robust compressive summarization with dual decomposition and multi-task learning. In *Proceedings of ACL*, pages 196–206, August.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of ACL-HLT*, pages 481–490.

Ted Briscoe, John Carroll, and Rebecca Watson. 2006. The second release of the RASP system. In *Proceedings of the ACL-COLING Interactive Presentation Sessions*.

Yoeng-jin Chu and Tseng-hong Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

James Clarke and Mirella Lapata. 2006. Models for sentence compression: a comparison across domains, training requirements and evaluation measures. In *Proceedings of ACL-COLING*, pages 377–384.

James Clarke and Mirella Lapata. 2007. Modelling compression with discourse constraints. In *Proceedings of EMNLP-CoNLL*, pages 1–11.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: an integer linear programming approach. *Journal for Artificial Intelligence Research*, 31:399–429, March.

Trevor Cohn and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *Proceedings of COLING*, pages 137–144.

Trevor Cohn and Mirella Lapata. 2009. Sentence compression as tree transduction. *Journal of Artificial Intelligence Research*, 34(1):637–674, April.

Michael Collins. 2002. Discriminative training methods for hidden Markov models. In *Proceedings of EMNLP*, pages 1–8.

Dipanjan Das, André F. T. Martins, and Noah A. Smith. 2012. An exact dual decomposition algorithm for shallow semantic parsing with constraints. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics (*SEM)*, SemEval '12, pages 209–217.

Hal Daumé, III and Daniel Marcu. 2002. A noisy-channel model for document compression. In *Proceedings of ACL*, pages 449–456.

John DeNero and Klaus Macherey. 2011. Model-based aligner combination using dual decomposition. In *Proceedings of ACL-HLT*, pages 420–429.

Jack R. Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of EMNLP*, pages 1481–1491.

Katja Filippova and Michael Strube. 2008. Dependency tree based sentence compression. In *Proceedings of INLG*, pages 25–32.

Dimitrios Galanis and Ion Androutsopoulos. 2010. An extractive supervised two-stage method for sentence compression. In *Proceedings of HLT-NAACL*, pages 885–893.

Michel Galley and Kathleen McKeown. 2007. Lexicalized Markov grammars for sentence compression. In *Proceedings of HLT-NAACL*, pages 180–187, April.

Juri Ganitkevitch, Chris Callison-Burch, Courtney Napoles, and Benjamin Van Durme. 2011. Learning sentential paraphrases from bilingual parallel corpora for text-to-text generation. In *Proceedings of EMNLP*, pages 1168–1179.

Chiori Hori and Sadaoki Furui. 2004. Speech summarization: an approach through word extraction and a method for evaluation. *IEICE Transactions on Information and Systems*, E87-D(1):15–25.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of ACL*, pages 144–151, June.

Hongyan Jing and Kathleen R. McKeown. 2000. Cut and paste based text summarization. In *Proceedings of NAACL*, pages 178–185.

Hongyan Jing. 2000. Sentence reduction for automatic text summarization. In *Proceedings of the Conference on Applied Natural Language Processing*, pages 310–315.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization - step one: Sentence compression. In *Proceedings of AAAI*, pages 703–710.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: a probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107, July.

Nikos Komodakis, Nikos Paragios, and Georgios Tziritas. 2007. MRF optimization via dual decomposition: Message-passing revisited. In *Proceedings of ICCV*, pages 1–8, Oct.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of EMNLP*, pages 1288–1298.

Alex Kulesza and Fernando Pereira. 2007. Structured learning with approximate inference. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *NIPS*. Curran Associates, Inc.

Hoong Chuin Lau, Trung Hieu Ngo, and Bao Nguyen Nguyen. 2006. Finding a length-constrained maximum-sum or maximum-density subtree and its application to logistics. *Discrete Optimization*, 3(4):385 – 391.

Chen Li, Fei Liu, Fuliang Weng, and Yang Liu. 2013. Document summarization via guided sentence compression. In *Proceedings of EMNLP*, pages 490–500, Seattle, Washington, USA, October.

Thomas L. Magnanti and Laurence A. Wolsey. 1994. Optimal trees. In *Technical Report 290-94, Massechusetts Institute of Technology, Operations Research Center*.

André F. T. Martins and Noah A. Smith. 2009. Summarization with a joint model for sentence extraction and compression. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, pages 1–9.

André F. T. Martins, Noah A. Smith, and Eric P. Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of ACL-IJCNLP*, pages 342–350.

André F. T. Martins, Noah A. Smith, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. 2011. Dual decomposition with many overlapping components. In *Proceedings of EMNLP*, pages 238–249.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of EMNLP-HLT*, pages 523–530.

Ryan McDonald. 2006. Discriminative sentence compression with soft syntactic evidence. In *Proceedings of EACL*, pages 297–304.

Alejandro Molina, Juan-Manuel Torres-Moreno, Eric SanJuan, Iria da Cunha, and Gerardo Eugenio Sierra Martínez. 2013. Discursive sentence compression. In *Computational Linguistics and Intelligent Text Processing*, volume 7817, pages 394–407. Springer.

Courtney Napoles, Chris Callison-Burch, Juri Ganitkevitch, and Benjamin Van Durme. 2011a. Paraphrastic sentence compression with a character-based metric: tightening without deletion. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 84–90.

Courtney Napoles, Benjamin Van Durme, and Chris Callison-Burch. 2011b. Evaluating sentence compression: pitfalls and suggested remedies. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 91–97.

Tadashi Nomoto. 2007. Discriminative sentence compression with conditional random fields. *Information Processing and Management*, 43(6):1571–1587, November.

Xian Qian and Yang Liu. 2013. Fast joint compression and summarization via graph cuts. In *Proceedings of EMNLP*, pages 1492–1502, Seattle, Washington, USA, October.

Stefan Riezler, Tracy H. King, Richard Crouch, and Annie Zaenen. 2003. Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for lexical-functional grammar. In *Proceedings of HLT-NAACL*, pages 118–125.

Alexander M. Rush and Michael Collins. 2011. Exact decoding of syntactic translation models through Lagrangian relaxation. In *Proceedings of ACL-HLT*, pages 72–82.

Alexander M. Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of EMNLP*, pages 1–11.

Kapil Thadani and Kathleen McKeown. 2013. Sentence compression with joint structural inference. In *Proceedings of CoNLL*.

Jenine Turner and Eugene Charniak. 2005. Supervised and unsupervised learning for sentence compression. In *Proceedings of ACL*, pages 290–297.

Yuya Unno, Takashi Ninomiya, Yusuke Miyao, and Jun'ichi Tsujii. 2006. Trimming CFG parse trees for sentence compression using machine learning approaches. In *Proceedings of ACL-COLING*, pages 850–857.

Kristian Woodsend and Mirella Lapata. 2012. Multiple aspect summarization using integer linear programming. In *Proceedings of EMNLP*, pages 233–243.

David Zajic, Bonnie J. Dorr, Jimmy Lin, and Richard Schwartz. 2007. Multi-candidate reduction: Sentence compression as a tool for document summarization tasks. *Information Processing and Management*, 43(6):1549–1570, Nov.

Kai Zhao and Liang Huang. 2013. Minibatch and parallelization for online large margin structured learning. In *Proceedings of HLT-NAACL*, pages 370–379, Atlanta, Georgia, June.

Opinion Mining on YouTube

Aliaksei Severyn[1], **Alessandro Moschitti**[3,1],
Olga Uryupina[1], **Barbara Plank**[2], **Katja Filippova**[4]
[1]DISI - University of Trento, [2]CLT - University of Copenhagen,
[3]Qatar Computing Research Institute, [4]Google Inc.
severyn@disi.unitn.it, amoschitti@qf.org.qa,
uryupina@gmail.com, bplank@cst.dk, katjaf@google.com

Abstract

This paper defines a systematic approach to Opinion Mining (OM) on YouTube comments by (i) modeling classifiers for predicting the opinion polarity and the type of comment and (ii) proposing robust shallow syntactic structures for improving model adaptability. We rely on the tree kernel technology to automatically extract and learn features with better generalization power than bag-of-words. An extensive empirical evaluation on our manually annotated YouTube comments corpus shows a high classification accuracy and highlights the benefits of structural models in a cross-domain setting.

1 Introduction

Social media such as Twitter, Facebook or YouTube contain rapidly changing information generated by millions of users that can dramatically affect the reputation of a person or an organization. This raises the importance of automatic extraction of sentiments and opinions expressed in social media.

YouTube is a unique environment, just like Twitter, but probably even richer: multi-modal, with a social graph, and discussions between people sharing an interest. Hence, doing sentiment research in such an environment is highly relevant for the community. While the linguistic conventions used on Twitter and YouTube indeed show similarities (Baldwin et al., 2013), focusing on YouTube allows to exploit context information, possibly also multi-modal information, not available in isolated tweets, thus rendering it a valuable resource for the future research.

Nevertheless, there is almost no work showing effective OM on YouTube comments. To the best of our knowledge, the only exception is given by the classification system of YouTube comments proposed by Siersdorfer et al. (2010).

While previous state-of-the-art models for opinion classification have been successfully applied to traditional corpora (Pang and Lee, 2008), YouTube comments pose additional challenges: (i) polarity words can refer to either video or product while expressing contrasting sentiments; (ii) many comments are unrelated or contain spam; and (iii) learning supervised models requires training data for each different YouTube domain, e.g., *tablets*, *automobiles*, etc. For example, consider a typical comment on a YouTube review video about a *Motorola Xoom* tablet:

> *this guy really puts a negative spin on this , and I 'm not sure why , this seems crazy fast , and I 'm not entirely sure why his pinch to zoom his laggy all the other* ***xoom*** *reviews*

The comment contains a product name *xoom* and some negative expressions, thus, a bag-of-words model would derive a negative polarity for this product. In contrast, the opinion towards the product is neutral as the negative sentiment is expressed towards the video. Similarly, the following comment:

> *iPad 2 is better. the superior apps just destroy the* ***xoom***.

contains two positive and one negative word, yet the sentiment towards the product is negative (the negative word *destroy* refers to *Xoom*). Clearly, the bag-of-words lacks the structural information linking the sentiment with the target product.

In this paper, we carry out a systematic study on OM targeting YouTube comments; its contribution is three-fold: firstly, to solve the problems outlined above, we define a classification schema, which separates spam and not related comments from the informative ones, which are, in turn, further categorized into video- or product-related comments

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1252–1261,
Baltimore, Maryland, USA, June 23-25 2014.

(type classification). At the final stage, different classifiers assign polarity (positive, negative or neutral) to each type of a meaningful comment. This allows us to filter out irrelevant comments, providing accurate OM distinguishing comments about the video and the target product.

The second contribution of the paper is the creation and annotation (by an expert coder) of a comment corpus containing 35k manually labeled comments for two product YouTube domains: *tablets* and *automobiles*.[1] It is the first manually annotated corpus that enables researchers to use supervised methods on YouTube for comment classification and opinion analysis. The comments from different product domains exhibit different properties (cf. Sec. 5.2), which give the possibility to study the domain adaptability of the supervised models by training on one category and testing on the other (and vice versa).

The third contribution of the paper is a novel structural representation, based on shallow syntactic trees enriched with conceptual information, i.e., tags generalizing the specific topic of the video, e.g., *iPad*, *Kindle*, *Toyota Camry*. Given the complexity and the novelty of the task, we exploit structural kernels to automatically engineer novel features. In particular, we define an efficient tree kernel derived from the Partial Tree Kernel, (Moschitti, 2006a), suitable for encoding structural representation of comments into Support Vector Machines (SVMs). Finally, our results show that our models are adaptable, especially when the structural information is used. Structural models generally improve on both tasks – polarity and type classification – yielding up to 30% of relative improvement, when little data is available. Hence, the impractical task of annotating data for each YouTube category can be mitigated by the use of models that adapt better across domains.

2 Related work

Most prior work on more general OM has been carried out on more standardized forms of text, such as consumer reviews or newswire. The most commonly used datasets include: the MPQA corpus of news documents (Wilson et al., 2005), web customer review data (Hu and Liu, 2004), Amazon review data (Blitzer et al., 2007), the JDPA corpus of blogs (Kessler et al., 2010), etc. The aforementioned corpora are, however, only partially suitable for developing models on social media, since the informal text poses additional challenges for Information Extraction and Natural Language Processing. Similar to Twitter, most YouTube comments are very short, the language is informal with numerous accidental and deliberate errors and grammatical inconsistencies, which makes previous corpora less suitable to train models for OM on YouTube. A recent study focuses on sentiment analysis for Twitter (Pak and Paroubek, 2010), however, their corpus was compiled automatically by searching for emoticons expressing positive and negative sentiment only.

Siersdorfer et al. (2010) focus on exploiting user ratings (counts of 'thumbs up/down' as flagged by other users) of YouTube video comments to train classifiers to predict the community acceptance of new comments. Hence, their goal is different: predicting comment ratings, rather than predicting the sentiment expressed in a YouTube comment or its information content. Exploiting the information from user ratings is a feature that we have not exploited thus far, but we believe that it is a valuable feature to use in future work.

Most of the previous work on supervised sentiment analysis use feature vectors to encode documents. While a few successful attempts have been made to use more involved linguistic analysis for opinion mining, such as dependency trees with latent nodes (Täckström and McDonald, 2011) and syntactic parse trees with vectorized nodes (Socher et al., 2011), recently, a comprehensive study by Wang and Manning (2012) showed that a simple model using bigrams and SVMs performs on par with more complex models.

In contrast, we show that adding structural features from syntactic trees is particularly useful for the cross-domain setting. They help to build a system that is more robust across domains. Therefore, rather than trying to build a specialized system for every new target domain, as it has been done in most prior work on domain adaptation (Blitzer et al., 2007; Daumé, 2007), the domain adaptation problem boils down to finding a more robust system (Søgaard and Johannsen, 2012; Plank and Moschitti, 2013). This is in line with recent advances in parsing the web (Petrov and McDonald, 2012), where participants where asked to build a single system able to cope with different yet re-

[1] The corpus and the annotation guidelines are publicly available at: `http://projects.disi.unitn.it/iKernels/projects/sentube/`

lated domains.

Our approach relies on robust syntactic structures to automatically generate patterns that adapt better. These representations have been inspired by the semantic models developed for Question Answering (Moschitti, 2008; Severyn and Moschitti, 2012; Severyn and Moschitti, 2013) and Semantic Textual Similarity (Severyn et al., 2013). Moreover, we introduce additional tags, e.g., video concepts, polarity and negation words, to achieve better generalization across different domains where the word distribution and vocabulary changes.

3 Representations and models

Our approach to OM on YouTube relies on the design of classifiers to predict comment type and opinion polarity. Such classifiers are traditionally based on bag-of-words and more advanced features. In the next sections, we define a baseline feature vector model and a novel structural model based on kernel methods.

3.1 Feature Set

We enrich the traditional bag-of-word representation with features from a sentiment lexicon and features quantifying the negation present in the comment. Our model (`FVEC`) encodes each document using the following feature groups:

- **word n-grams**: we compute unigrams and bigrams over lower-cased word lemmas where binary values are used to indicate the presence/absence of a given item.
- **lexicon**: a sentiment lexicon is a collection of words associated with a positive or negative sentiment. We use two manually constructed sentiment lexicons that are freely available: the MPQA Lexicon (Wilson et al., 2005) and the lexicon of Hu and Liu (2004). For each of the lexicons, we use the number of words found in the comment that have *positive* and *negative* sentiment as a feature.
- **negation**: the count of negation words, e.g., {*don't, never, not, etc.*}, found in a comment.[2] Our structural representation (defined next) enables a more involved treatment of negation.
- **video concept**: cosine similarity between a comment and the title/description of the video. Most of the videos come with a title and a short description, which can be used to encode the topicality of each comment by looking at their overlap.

3.2 Structural model

We go beyond traditional feature vectors by employing structural models (`STRUCT`), which encode each comment into a shallow syntactic tree. These trees are input to tree kernel functions for generating structural features. Our structures are specifically adapted to the noisy user-generated texts and encode important aspects of the comments, e.g., words from the sentiment lexicons, product concepts and negation words, which specifically targets the sentiment and comment type classification tasks.

In particular, our shallow tree structure is a two-level syntactic hierarchy built from word lemmas (leaves) and part-of-speech tags that are further grouped into chunks (Fig. 1). As full syntactic parsers such as constituency or dependency tree parsers would significantly degrade in performance on noisy texts, e.g., Twitter or YouTube comments, we opted for shallow structures, which rely on simpler and more robust components: a part-of-speech tagger and a chunker. Moreover, such taggers have been recently updated with models (Ritter et al., 2011; Gimpel et al., 2011) trained specifically to process noisy texts showing significant reductions in the error rate on user-generated texts, e.g., Twitter. Hence, we use the CMU Twitter pos-tagger (Gimpel et al., 2011; Owoputi et al., 2013) to obtain the part-of-speech tags. Our second component – chunker – is taken from (Ritter et al., 2011), which also comes with a model trained on Twitter data[3] and shown to perform better on noisy data such as user comments.

To address the specifics of OM tasks on YouTube comments, we enrich syntactic trees with semantic tags to encode: (i) central concepts of the video, (ii) sentiment-bearing words expressing *positive* or *negative* sentiment and (iii) negation words. To automatically identify concept words of the video we use context words (tokens detected as nouns by the part-of-speech tagger) from the video title and video description and match them in the tree. For the matched words, we enrich labels of their parent nodes (part-of-speech and chunk) with the `PRODUCT` tag. Similarly, the nodes associated with words found in

[2] The list of negation words is adopted from http://sentiment.christopherpotts.net/lingstruc.html

[3] The chunker from (Ritter et al., 2011) relies on its own POS tagger, however, in our structural representations we favor the POS tags from the CMU Twitter tagger and take only the chunk tags from the chunker.

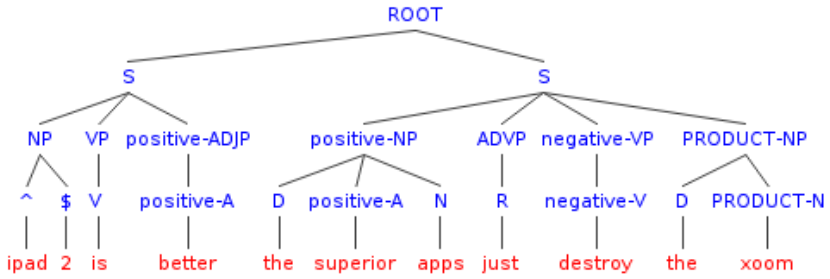


Figure 1: Shallow tree representation of the example comment (labeled with `product` type and `negative` sentiment): *"iPad 2 is better. the superior apps just destroy the xoom."* (lemmas are replaced with words for readability) taken from the video "Motorola Xoom Review". We introduce additional tags in the tree nodes to encode the central concept of the video (motorola *xoom*) and sentiment-bearing words *(better, superior, destroy)* directly in the tree nodes. For the former we add a `PRODUCT` tag on the chunk and part-of-speech nodes of the word *xoom*) and polarity tags (*positive* and *negative*) for the latter. Two sentences are split into separate root nodes `S`.

the sentiment lexicon are enriched with a polarity tag (either *positive* or *negative*), while negation words are labeled with the `NEG` tag. It should be noted that vector-based (`FVEC`) model relies only on feature counts whereas the proposed tree encodes powerful contextual syntactic features in terms of tree fragments. The latter are automatically generated and learned by SVMs with expressive tree kernels.

For example, the comment in Figure 1 shows two *positive* and one *negative* word from the sentiment lexicon. This would strongly bias the `FVEC` sentiment classifier to assign a `positive` label to the comment. In contrast, the `STRUCT` model relies on the fact that the negative word, *destroy*, refers to the PRODUCT (*xoom*) since they form a verbal phase (VP). In other words, the tree fragment: `[S [negative-VP [negative-V [destroy]] [PRODUCT-NP [PRODUCT-N [xoom]]]]` is a strong feature (induced by tree kernels) to help the classifier to discriminate such hard cases. Moreover, tree kernels generate all possible subtrees, thus producing generalized (back-off) features, e.g., `[S [negative-VP [negative-V [destroy]] [PRODUCT-NP]]]]` or `[S [negative-VP [PRODUCT-NP]]]]`.

3.3 Learning

We perform OM on YouTube using supervised methods, e.g., SVM. Our goal is to learn a model to automatically detect the sentiment and type of each comment. For this purpose, we build a multi-class classifier using the one-vs-all scheme. A binary classifier is trained for each of the classes and the predicted class is obtained by taking a class from the classifier with a maximum prediction score. Our back-end binary classifier is SVM-light-TK[4], which encodes structural kernels in the SVM-light (Joachims, 2002) solver. We define a novel and efficient tree kernel function, namely, **Sh**allow syntactic **T**ree **K**ernel (SHTK), which is as expressive as the Partial Tree Kernel (PTK) (Moschitti, 2006a) to handle feature engineering over the structural representations of the `STRUCT` model. A polynomial kernel of degree 3 is applied to feature vectors (`FVEC`).

Combining structural and vector models. A typical kernel machine, e.g., SVM, classifies a test input $\boldsymbol{x}$ using the following prediction function: $h(\boldsymbol{x}) = \sum_i \alpha_i y_i K(\boldsymbol{x}, \boldsymbol{x}_i)$, where α_i are the model parameters estimated from the training data, y_i are target variables, $\boldsymbol{x}_i$ are support vectors, and $K(\cdot,\cdot)$ is a kernel function. The latter computes the *similarity* between two comments. The `STRUCT` model treats each comment as a tuple $\boldsymbol{x} = \langle \boldsymbol{T}, \boldsymbol{v} \rangle$ composed of a shallow syntactic tree $\boldsymbol{T}$ and a feature vector $\boldsymbol{v}$. Hence, for each pair of comments $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, we define the following comment similarity kernel:

$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = K_{\text{TK}}(\boldsymbol{T}_1, \boldsymbol{T}_2) + K_{\text{v}}(\boldsymbol{v}_1, \boldsymbol{v}_2), \quad (1)$$

where K_{TK} computes SHTK (defined next), and K_{v} is a kernel over feature vectors, e.g., linear, polynomial, Gaussian, etc.

Shallow syntactic tree kernel. Following the convolution kernel framework, we define the new

[4] http://disi.unitn.it/moschitti/Tree-Kernel.htm

SHTK function from Eq. 1 to compute the similarity between tree structures. It counts the number of common substructures between two trees T_1 and T_2 without explicitly considering the whole fragment space. The general equations for Convolution Tree Kernels is:

$$TK(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2), \quad (2)$$

where N_{T_1} and N_{T_2} are the sets of the T_1's and T_2's nodes, respectively and $\Delta(n_1, n_2)$ is equal to the number of common fragments rooted in the n_1 and n_2 nodes, according to several possible definition of the atomic fragments.

To improve the speed computation of TK, we consider pairs of nodes (n_1, n_2) belonging to the same tree level. Thus, given H, the height of the `STRUCT` trees, where each level h contains nodes of the same type, i.e., chunk, POS, and lexical nodes, we define SHTK as the following[5]:

$$SHTK(T_1, T_2) = \sum_{h=1}^{H} \sum_{n_1 \in N_{T_1}^h} \sum_{n_2 \in N_{T_2}^h} \Delta(n_1, n_2), \quad (3)$$

where $N_{T_1}^h$ and $N_{T_2}^h$ are sets of nodes at height h.

The above equation can be applied with any Δ function. To have a more general and expressive kernel, we use Δ previously defined for PTK. More formally: if n_1 and n_2 are leaves then $\Delta(n_1, n_2) = \mu\lambda(n_1, n_2)$; else $\Delta(n_1, n_2) =$

$$\mu\Big(\lambda^2 + \sum_{\vec{I}_1, \vec{I}_2, |\vec{I}_1| = |\vec{I}_2|} \lambda^{d(\vec{I}_1) + d(\vec{I}_2)} \prod_{j=1}^{|\vec{I}_1|} \Delta(c_{n_1}(\vec{I}_{1j}), c_{n_2}(\vec{I}_{2j}))\Big),$$

where $\lambda, \mu \in [0, 1]$ are decay factors; the large sum is adopted from a definition of the subsequence kernel (Shawe-Taylor and Cristianini, 2004) to generate children subsets with gaps, which are then used in a recursive call to Δ. Here, $c_{n_1}(i)$ is the i^{th} child of the node n_1; $\vec{I}_1$ and $\vec{I}_2$ are two sequences of indexes that enumerate subsets of children with gaps, i.e., $\vec{I} = (i_1, i_2, .., |I|)$, with $1 \leq i_1 < i_2 < .. < i_{|I|}$; and $d(\vec{I}_1) = \vec{I}_{1l(\vec{I}_1)} - \vec{I}_{11} + 1$ and $d(\vec{I}_2) = \vec{I}_{2l(\vec{I}_2)} - \vec{I}_{21} + 1$, which penalizes subsequences with larger gaps.

It should be noted that: firstly, the use of a subsequence kernel makes it possible to generate child subsets of the two nodes, i.e., it allows for gaps, which makes matching of syntactic patterns less rigid. Secondly, the resulting SHTK is essentially a special case of PTK (Moschitti, 2006a), adapted to the shallow structural representation `STRUCT` (see Sec. 3.2). When applied to `STRUCT` trees, SHTK exactly computes the same feature space as PTK, but in faster time (on average). Indeed, SHTK required to be only applied to node pairs from the same level (see Eq. 3), where the node labels can match – chunk, POS or lexicals. This reduces the time for selecting the matching-node pairs carried out in PTK (Moschitti, 2006a; Moschitti, 2006b). The fragment space is obviously the same, as the node labels of different levels in `STRUCT` are different and will not be matched by PTK either.

Finally, given its recursive definition in Eq. 3 and the use of subsequence (with gaps), SHTK can derive useful dependencies between its elements. For example, it will generate the following subtree fragments: `[positive-NP [positive-AN]]`, `[S [negative-VP [negative-V [destroy]] [PRODUCT-NP]]]` and so on.

4 YouTube comments corpus

To build a corpus of YouTube comments, we focus on a particular set of videos (technical reviews and advertisings) featuring commercial products. In particular, we chose two product categories: automobiles (`AUTO`) and tablets (`TABLETS`). To collect the videos, we compiled a list of products and queried the YouTube gData API[6] to retrieve the videos. We then manually excluded irrelevant videos. For each video, we extracted all available comments (limited to maximum 1k comments per video) and manually annotated each comment with its type and polarity. We distinguish between the following types:

product: discuss the topic product in general or some features of the product;

video: discuss the video or some of its details;

spam: provide advertising and malicious links; and

off-topic: comments that have almost no content ("lmao") or content that is not related to the video ("Thank you!").

Regarding the polarity, we distinguish between {*positive, negative, neutral*} sentiments with respect to the product and the video. If the comment contains several statements of different polarities, it is annotated as both *positive* and *negative*: "Love the video but waiting for iPad 4". In total we have

[5] To have a similarity score between 0 and 1, a normalization in the kernel space, i.e. $\frac{SHTK(T_1,T_2)}{\sqrt{SHTK(T_1,T_1) \times SHTK(T_2,T_2)}}$ is applied.

[6] https://developers.google.com/youtube/v3/

annotated 208 videos with around 35k comments (128 videos TABLETS and 80 for AUTO).

To evaluate the quality of the produced labels, we asked 5 annotators to label a sample set of one hundred comments and measured the agreement. The resulting annotator agreement α value (Krippendorf, 2004; Artstein and Poesio, 2008) scores are 60.6 (AUTO), 72.1 (TABLETS) for the sentiment task and 64.1 (AUTO), 79.3 (TABLETS) for the **type** classification task. For the rest of the comments, we assigned the entire annotation task to a single coder. Further details on the corpus can be found in Uryupina et al. (2014).

5 Experiments

This section reports: (i) experiments on individual subtasks of opinion and type classification; (ii) the full task of predicting type and sentiment; (iii) study on the adaptability of our system by learning on one domain and testing on the other; (iv) learning curves that provide an indication on the required amount and type of data and the scalability to other domains.

5.1 Task description

Sentiment classification. We treat each comment as expressing positive, negative or neutral sentiment. Hence, the task is a three-way classification.

Type classification. One of the challenging aspects of sentiment analysis of YouTube data is that the comments may express the sentiment not only towards the product shown in the video, but also the video itself, i.e., users may post positive comments to the video while being generally negative about the product and vice versa. Hence, it is of crucial importance to distinguish between these two types of comments. Additionally, many comments are irrelevant for both the product and the video (off-topic) or may even contain spam. Given that the main goal of sentiment analysis is to select sentiment-bearing comments and identify their polarity, distinguishing between off-topic and spam categories is not critical. Thus, we merge the spam and off-topic into a single uninformative category. Similar to the opinion classification task, comment type classification is a multi-class classification with three classes: video, product and uninform.

Full task. While the previously discussed sentiment and type identification tasks are useful to model and study in their own right, our end goal is: given a stream of comments, to jointly predict both the type and the sentiment of each comment. We cast this problem as a single multi-class classification task with seven classes: the Cartesian product between {product, video} type labels and {positive, neutral, negative} sentiment labels plus the uninformative category (*spam* and *off-topic*). Considering a real-life application, it is important not only to detect the polarity of the comment, but to also identify if it is expressed towards the product or the video.[7]

Task	class	AUTO		TABLETS	
		TRAIN	TEST	TRAIN	TEST
Sentiment	positive	2005 (36%)	807 (27%)	2393 (27%)	1872 (27%)
	neutral	2649 (48%)	1413 (47%)	4683 (53%)	3617 (52%)
	negative	878 (16%)	760 (26%)	1698 (19%)	1471 (21%)
	total	5532	2980	8774	6960
Type	product	2733 (33%)	1761 (34%)	7180 (59%)	5731 (61%)
	video	3008 (36%)	1369 (26%)	2088 (17%)	1674 (18%)
	off-topic	2638 (31%)	2045 (39%)	2334 (19%)	1606 (17%)
	spam	26 (>1%)	17 (>1%)	658 (5%)	361 (4%)
	total	8405	5192	12260	9372
Full	product-pos.	1096 (13%)	517 (10%)	1648 (14%)	1278 (14%)
	product-neu.	908 (11%)	729 (14%)	3681 (31%)	2844 (32%)
	product-neg.	554 (7%)	370 (7%)	1404 (12%)	1209 (14%)
	video-pos.	909 (11%)	290 (6%)	745 (6%)	594 (7%)
	video-neu.	1741 (21%)	683 (14%)	1002 (9%)	773 (9%)
	video-neg.	324 (4%)	390 (8%)	294 (2%)	262 (3%)
	off-topic	2638 (32%)	2045 (41%)	2334 (20%)	1606 (18%)
	spam	26 (>1%)	17 (>1%)	658 (6%)	361 (4%)
	total	8196	5041	11766	8927

Table 1: Summary of YouTube comments data used in the sentiment, type and full classification tasks. The comments come from two product categories: AUTO and TABLETS. Numbers in parenthesis show proportion w.r.t. to the total number of comments used in a task.

5.2 Data

We split all the videos 50% between training set (TRAIN) and test set (TEST), where each video contains all its comments. This ensures that all comments from the same video appear either in TRAIN or in TEST. Since the number of comments per video varies, the resulting sizes of each set are different (we use the larger split for TRAIN). Table 1 shows the data distribution across the task-specific classes – **sentiment** and **type** classification. For the **sentiment** task we exclude off-topic and spam comments as well as comments with ambiguous sentiment, i.e., an-

[7] We exclude comments annotated as both video and product. This enables the use of a simple flat multi-classifiers with seven categories for the full task, instead of a hierarchical multi-label classifiers (i.e., type classification first and then opinion polarity). The number of comments assigned to both product and video is relatively small (8% for TABLETS and 4% for AUTO).

notated as both positive and negative.

For the **sentiment** task about 50% of the comments have neutral polarity, while the negative class is much less frequent. Interestingly, the ratios between polarities expressed in comments from AUTO and TABLETS are very similar across both TRAIN and TEST. Conversely, for the **type** task, we observe that comments from AUTO are uniformly distributed among the three classes, while for the TABLETS the majority of comments are product related. It is likely due to the nature of the TABLETS videos, that are more geek-oriented, where users are more prone to share their opinions and enter involved discussions about a product. Additionally, videos from the AUTO category (both commercials and user reviews) are more visually captivating and, being generally oriented towards a larger audience, generate more video-related comments. Regarding the **full** setting, where the goal is to have a joint prediction of the comment sentiment and type, we observe that video-negative and video-positive are the most scarce classes, which makes them the most difficult to predict.

5.3 Results

We start off by presenting the results for the traditional in-domain setting, where both TRAIN and TEST come from the same domain, e.g., AUTO or TABLETS. Next, we show the learning curves to analyze the behavior of FVEC and STRUCT models according to the training size. Finally, we perform a set of cross-domain experiments that describe the enhanced adaptability of the patterns generated by the STRUCT model.

5.3.1 In-domain experiments

We compare FVEC and STRUCT models on three tasks described in Sec. 5.1: sentiment, type and full. Table 2 reports the per-class performance and the overall accuracy of the multi-class classifier. Firstly, we note that the performance on TABLETS is much higher than on AUTO across all tasks. This can be explained by the following: (i) TABLETS contains more training data and (ii) videos from AUTO and TABLETS categories draw different types of audiences – well-informed users and geeks expressing better-motivated opinions about a product for the former vs. more general audience for the latter. This results in the different quality of comments with the AUTO being more challenging to analyze. Secondly, we observe that the STRUCT model provides 1-3% of absolute improvement in accuracy over FVEC for every task. For individual categories the F1 scores are also improved by the STRUCT model (except for the *negative* classes for AUTO, where we see a small drop). We conjecture that sentiment prediction for AUTO category is largely driven by one-shot phrases and statements where it is hard to improve upon the bag-of-words and sentiment lexicon features. In contrast, comments from TABLETS category tend to be more elaborated and well-argumented, thus, benefiting from the expressiveness of the structural representations.

Considering per-class performance, correctly predicting negative sentiment is most difficult for both AUTO and TABLETS, which is probably caused by the smaller proportion of the negative comments in the training set. For the **type** task, video-related class is substantially more difficult than product-related for both categories. For the **full** task, the class video-negative accounts for the largest error. This is confirmed by the results from the previous sentiment and type tasks, where we saw that handling negative sentiment and detecting video-related comments are most difficult.

5.3.2 Learning curves

The learning curves depict the behavior of FVEC and STRUCT models as we increase the size of the training set. Intuitively, the STRUCT model relies on more general syntactic patterns and may overcome the sparseness problems incurred by the FVEC model when little training data is available.

Nevertheless, as we see in Figure 2, the learning curves for sentiment and type classification tasks across both product categories do not confirm this intuition. The STRUCT model consistently outperforms the FVEC across all training sizes, but the gap in the performance does not increase when we move to smaller training sets. As we will see next, this picture changes when we perform the cross-domain study.

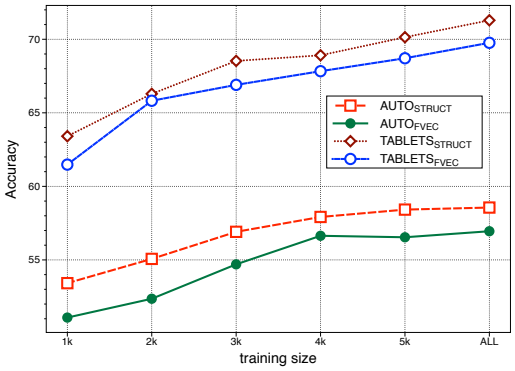
5.3.3 Cross-domain experiments

To understand the performance of our classifiers on other YouTube domains, we perform a set of cross-domain experiments by training on the data from one product category and testing on the other.

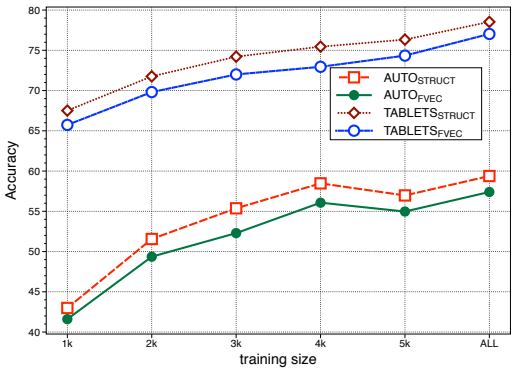
Table 3 reports the accuracy for three tasks when we use all comments (TRAIN + TEST) from AUTO to predict on the TEST from TABLETS

Task	class	AUTO						TABLETS					
		FVEC			STRUCT			FVEC			STRUCT		
		P	R	F1	P	R	F1	P	R	F1	P	R	F1
Sent	positive	49.1	72.1	58.4	50.1	73.9	59.0	67.5	70.3	69.9	71.2	71.3	71.3
	neutral	68.2	55.0	61.4	70.1	57.6	63.1	81.3	71.4	76.9	81.1	73.1	77.8
	negative	42.0	36.9	39.6	41.3	35.8	38.8	48.3	60.0	54.8	50.2	62.6	56.5
	Acc			54.7			55.7			68.6			70.5
Type	product	66.8	73.3	69.4	68.8	75.5	71.7	78.2	95.3	86.4	80.1	95.5	87.6
	video	45.0	52.8	48.2	47.8	49.9	48.7	83.6	45.7	58.9	83.5	46.7	59.4
	uninform	59.3	48.2	53.1	60.6	53.0	56.4	70.2	52.5	60.7	72.9	58.6	65.0
	Acc			57.4			59.4			77.2			78.6
Full	product-pos	34.0	49.6	39.2	36.5	51.2	43.0	48.4	56.8	52.0	52.4	59.3	56.4
	product-neu	43.4	31.1	36.1	41.4	36.1	38.4	68.0	67.5	68.1	59.7	83.4	70.0
	product-neg	26.3	29.5	28.8	26.3	25.3	25.6	43.0	49.9	45.4	44.7	53.7	48.4
	video-pos	23.2	47.1	31.9	26.1	54.5	35.5	69.1	60.0	64.7	64.9	68.8	66.4
	video-neu	26.1	30.0	29.0	26.5	31.6	28.8	56.4	32.1	40.0	55.1	35.7	43.3
	video-neg	21.9	3.7	6.0	17.7	2.3	4.8	39.0	17.5	23.9	39.5	6.1	11.5
	uninform	56.5	52.4	54.9	60.0	53.3	56.3	60.0	65.5	62.2	63.3	68.4	66.9
	Acc			40.0			41.5			57.6			60.3

Table 2: In-domain experiments on AUTO and TABLETS using two models: FVEC and STRUCT. The results are reported for sentiment, type and full classification tasks. The metrics used are precision (P), recall (R) and F1 for each individual class and the general accuracy of the multi-class classifier (Acc).



(a) Sentiment classification



(b) Type classification

Figure 2: In-domain learning curves. ALL refers to the entire TRAIN set for a given product category, i.e., AUTO and TABLETS (see Table 1)

and in the opposite direction (TABLETS→AUTO). When using AUTO as a source domain, STRUCT model provides additional 1-3% of absolute improvement, except for the sentiment task.

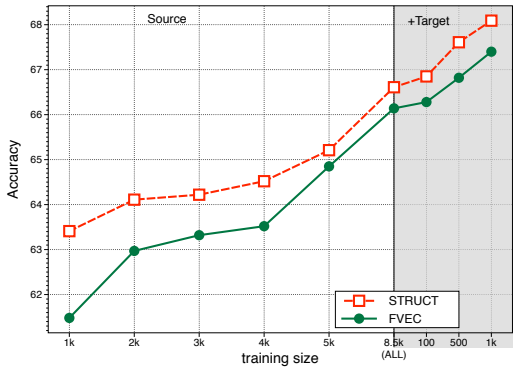
Source	Target	Task	FVEC	STRUCT
AUTO	TABLETS	Sent	66.1	66.6
		Type	59.9	64.1†
		Full	35.6	38.3†
TABLETS	AUTO	Sent	60.4	61.9†
		Type	54.2	55.6†
		Full	43.4	44.7†

Table 3: Cross-domain experiment. Accuracy using FVEC and STRUCT models when trained/tested in both directions, i.e. AUTO→TABLETS and TABLETS→AUTO. † denotes results statistically significant at 95% level (via pairwise t-test).

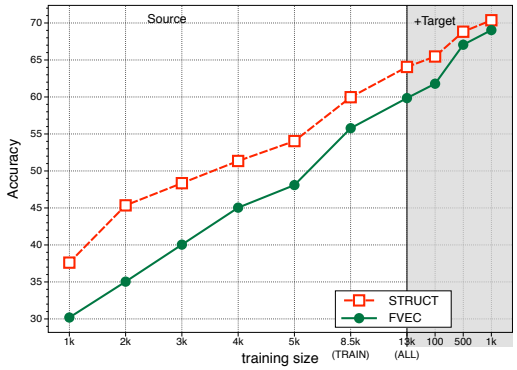
Similar to the in-domain experiments, we studied the effect of the source domain size on the target test performance. This is useful to assess the adaptability of features exploited by the FVEC and STRUCT models with the change in the number of labeled examples available for training. Additionally, we considered a setting including a small amount of training data from the target data (i.e., supervised domain adaptation).

For this purpose, we drew the learning curves of the FVEC and STRUCT models applied to the **sentiment** and **type** tasks (Figure 3): AUTO is used as the source domain to train models, which are tested on TABLETS.[8] The plot shows that when

[8]The results for the other direction (TABLETS→AUTO) show similar behavior.



(a) Sentiment classification



(b) Type classification

Figure 3: Learning curves for the cross-domain setting (AUTO→TABLETS). Shaded area refers to adding a small portion of comments from the same domain as the target test data to the training.

little training data is available, the features generated by the STRUCT model exhibit better adaptability (up to 10% of improvement over FVEC). The bag-of-words model seems to be affected by the data sparsity problem which becomes a crucial issue when only a small training set is available. This difference becomes smaller as we add data from the same domain. This is an important advantage of our structural approach, since we cannot realistically expect to obtain manual annotations for 10k+ comments for each (of many thousands) product domains present on YouTube.

5.4 Discussion

Our STRUCT model is more accurate since it is able to induce structural patterns of sentiment. Consider the following comment: *optimus pad is better. this xoom is just to bulky but optimus pad offers better functionality*. The FVEC bag-of-words model misclassifies it to be positive, since it contains two positive expressions (*better*, *better functionality*) that outweigh a single negative expression (*bulky*). The structural model, in contrast, is able to identify the product of interest (*xoom*) and associate it with the negative expression through a structural feature and thus correctly classify the comment as negative.

Some issues remain problematic even for the structural model. The largest group of errors are implicit sentiments. Thus, some comments do not contain any explicit positive or negative opinions, but provide detailed and well-argumented criticism, for example, *this phone is heavy*. Such comments might also include irony. To account for these cases, a deep understanding of the product domain is necessary.

6 Conclusions and Future Work

We carried out a systematic study on OM from YouTube comments by training a set of supervised multi-class classifiers distinguishing between video and product related opinions. We use standard feature vectors augmented by shallow syntactic trees enriched with additional conceptual information.

This paper makes several contributions: (i) it shows that effective OM can be carried out with supervised models trained on high quality annotations; (ii) it introduces a novel annotated corpus of YouTube comments, which we make available for the research community; (iii) it defines novel structural models and kernels, which can improve on feature vectors, e.g., up to 30% of relative improvement in type classification, when little data is available, and demonstrates that the structural model scales well to other domains.

In the future, we plan to work on a joint model to classify all the comments of a given video, s.t. it is possible to exploit latent dependencies between entities and the sentiments of the comment thread. Additionally, we plan to experiment with hierarchical multi-label classifiers for the full task (in place of a flat multi-class learner).

Acknowledgments

The authors are supported by a Google Faculty Award 2011, the Google Europe Fellowship Award 2013 and the European Community's Seventh Framework Programme (FP7/2007-2013) under the grant #288024: LIMOSINE.

References

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596, December.

Timothy Baldwin, Paul Cook, Marco Lui, Andrew MacKinlay, and Li Wang. 2013. How noisy social media text, how diffrnt social media sources? In *IJCNLP*.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*.

Hal Daumé, III. 2007. Frustratingly easy domain adaptation. *ACL*.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for Twitter: annotation, features, and experiments. In *ACL*.

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

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *KDD*.

Jason S. Kessler, Miriam Eckert, Lyndsie Clark, and Nicolas Nicolov. 2010. The 2010 ICWSM JDPA sentiment corpus for the automotive domain. In *ICWSM-DWC*.

Klaus Krippendorf, 2004. *Content Analysis: An Introduction to Its Methodology, second edition*, chapter 11. Sage, Thousand Oaks, CA.

Alessandro Moschitti. 2006a. Efficient convolution kernels for dependency and constituent syntactic trees. In *ECML*.

Alessandro Moschitti. 2006b. Making tree kernels practical for natural language learning. In *EACL*, pages 113–120.

Alessandro Moschitti. 2008. Kernel methods, syntax and semantics for relational text categorization. In *CIKM*.

Olutobi Owoputi, Brendan OConnor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of NAACL-HLT*.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *LREC*.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *ACL*.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: an experimental study. In *ACL*.

Aliaksei Severyn and Alessandro Moschitti. 2012. Structural relationships for large-scale learning of answer re-ranking. In *SIGIR*.

Aliaksei Severyn and Alessandro Moschitti. 2013. Automatic feature engineering for answer selection and extraction. In *EMNLP*.

Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. 2013. Learning semantic textual similarity with structural representations. In *ACL*.

John Shawe-Taylor and Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Stefan Siersdorfer, Sergiu Chelaru, Wolfgang Nejdl, and Jose San Pedro. 2010. How useful are your comments?: Analyzing and predicting YouTube comments and comment ratings. In *WWW*.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *EMNLP*.

Anders Søgaard and Anders Johannsen. 2012. Robust learning in random subspaces: Equipping nlp for oov effects. In *COLING*.

Oscar Täckström and Ryan McDonald. 2011. Semi-supervised latent variable models for sentence-level sentiment analysis. In *ACL*.

Olga Uryupina, Barbara Plank, Aliaksei Severyn, Agata Rotondi, and Alessandro Moschitti. 2014. SenTube: A corpus for sentiment analysis on YouTube social media. In *LREC*.

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

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *EMNLP*.

Automatic Keyphrase Extraction: A Survey of the State of the Art

Kazi Saidul Hasan and **Vincent Ng**
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
{saidul,vince}@hlt.utdallas.edu

Abstract

While automatic keyphrase extraction has been examined extensively, state-of-the-art performance on this task is still much lower than that on many core natural language processing tasks. We present a survey of the state of the art in automatic keyphrase extraction, examining the major sources of errors made by existing systems and discussing the challenges ahead.

1 Introduction

Automatic keyphrase extraction concerns "the automatic selection of important and topical phrases from the body of a document" (Turney, 2000). In other words, its goal is to extract a set of phrases that are related to the main topics discussed in a given document (Tomokiyo and Hurst, 2003; Liu et al., 2009b; Ding et al., 2011; Zhao et al., 2011).

Document keyphrases have enabled fast and accurate searching for a given document from a large text collection, and have exhibited their potential in improving many natural language processing (NLP) and information retrieval (IR) tasks, such as text summarization (Zhang et al., 2004), text categorization (Hulth and Megyesi, 2006), opinion mining (Berend, 2011), and document indexing (Gutwin et al., 1999).

Owing to its importance, automatic keyphrase extraction has received a lot of attention. However, the task is far from being solved: state-of-the-art performance on keyphrase extraction is still much lower than that on many core NLP tasks (Liu et al., 2010). Our goal in this paper is to survey the state of the art in keyphrase extraction, examining the major sources of errors made by existing systems and discussing the challenges ahead.

2 Corpora

Automatic keyphrase extraction systems have been evaluated on corpora from a variety of sources ranging from long scientific publications to short paper abstracts and email messages. Table 1 presents a listing of the corpora grouped by their sources as well as their statistics.[1] There are at least four corpus-related factors that affect the difficulty of keyphrase extraction.

Length The difficulty of the task increases with the length of the input document as longer documents yield more candidate keyphrases (i.e., phrases that are eligible to be keyphrases (see Section 3.1)). For instance, each *Inspec* abstract has on average 10 annotator-assigned keyphrases and 34 candidate keyphrases. In contrast, a scientific paper typically has at least 10 keyphrases and hundreds of candidate keyphrases, yielding a much bigger search space (Hasan and Ng, 2010). Consequently, it is harder to extract keyphrases from scientific papers, technical reports, and meeting transcripts than abstracts, emails, and news articles.

Structural consistency In a structured document, there are certain locations where a keyphrase is most likely to appear. For instance, most of a scientific paper's keyphrases should appear in the abstract and the introduction. While structural information has been exploited to extract keyphrases from scientific papers (e.g., title, section information) (Kim et al., 2013), web pages (e.g., metadata) (Yih et al., 2006), and chats (e.g., dialogue acts) (Kim and Baldwin, 2012), it is most useful when the documents from a source exhibit structural similarity. For this reason, structural information is likely to facilitate keyphrase extraction from scientific papers and technical reports because of their standard format (i.e., standard sections such as abstract, introduction, conclusion, etc.). In contrast, the lack of structural consistency in other types of structured documents (e.g., web pages, which can be blogs, forums, or reviews)

[1] Many of the publicly available corpora can be found in http://github.com/snkim/AutomaticKeyphraseExtraction/ and http://code.google.com/p/maui-indexer/downloads/list.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1262–1273,
Baltimore, Maryland, USA, June 23-25 2014.

Source	Dataset/Contributor	Statistics		
		Documents	Tokens/doc	Keys/doc
Paper abstracts	*Inspec* (Hulth, 2003)∗	2,000	<200	10
Scientific papers	NUS corpus (Nguyen and Kan, 2007)∗	211	≈8K	11
	citeulike.org (Medelyan et al., 2009)∗	180	-	5
	SemEval-2010 (Kim et al., 2010b)∗	284	>5K	15
Technical reports	NZDL (Witten et al., 1999)∗	1,800	-	-
News articles	DUC-2001 (Wan and Xiao, 2008b)∗	308	≈900	8
	Reuters corpus (Hulth and Megyesi, 2006)	12,848	-	6
Web pages	Yih et al. (2006)	828	-	-
	Hammouda et al. (2005)∗	312	≈500	-
	Blogs (Grineva et al., 2009)	252	≈1K	8
Meeting transcripts	ICSI (Liu et al., 2009a)	161	≈1.6K	4
Emails	Enron corpus (Dredze et al., 2008)∗	14,659	-	-
Live chats	Library of Congress (Kim and Baldwin, 2012)	15	-	10

Table 1: Evaluation datasets. Publicly available datasets are marked with an asterisk (∗).

may render structural information less useful.

Topic change An observation commonly exploited in keyphrase extraction from scientific articles and news articles is that keyphrases typically appear not only at the beginning (Witten et al., 1999) but also at the end (Medelyan et al., 2009) of a document. This observation does not necessarily hold for conversational text (e.g., meetings, chats), however. The reason is simple: in a conversation, the topics (i.e., its talking points) change as the interaction moves forward in time, and so do the keyphrases associated with a topic. One way to address this complication is to detect a topic change in conversational text (Kim and Baldwin, 2012). However, topic change detection is not always easy: while the topics listed in the form of an agenda at the beginning of formal meeting transcripts can be exploited, such clues are absent in casual conversations (e.g., chats).

Topic correlation Another observation commonly exploited in keyphrase extraction from scientific articles and news articles is that the keyphrases in a document are typically *related* to each other (Turney, 2003; Mihalcea and Tarau, 2004). However, this observation does not necessarily hold for informal text (e.g., emails, chats, informal meetings, personal blogs), where people can talk about any number of potentially uncorrelated topics. The presence of uncorrelated topics implies that it may no longer be possible to exploit relatedness and therefore increases the difficulty of keyphrase extraction.

3 Keyphrase Extraction Approaches

A keyphrase extraction system typically operates in two steps: (1) extracting a list of words/phrases that serve as *candidate keyphrases* using some heuristics (Section 3.1); and (2) determining which of these candidate keyphrases are correct keyphrases using supervised (Section 3.2) or unsupervised (Section 3.3) approaches.

3.1 Selecting Candidate Words and Phrases

As noted before, a set of phrases and words is typically extracted as candidate keyphrases using heuristic rules. These rules are designed to avoid spurious instances and keep the number of candidates to a minimum. Typical heuristics include (1) using a stop word list to remove stop words (Liu et al., 2009b), (2) allowing words with certain part-of-speech tags (e.g., nouns, adjectives, verbs) to be candidate keywords (Mihalcea and Tarau, 2004; Wan and Xiao, 2008b; Liu et al., 2009a), (3) allowing n-grams that appear in Wikipedia article titles to be candidates (Grineva et al., 2009), and (4) extracting n-grams (Witten et al., 1999; Hulth, 2003; Medelyan et al., 2009) or noun phrases (Barker and Cornacchia, 2000; Wu et al., 2005) that satisfy pre-defined lexico-syntactic pattern(s) (Nguyen and Phan, 2009).

Many of these heuristics have proven effective with their high recall in extracting gold keyphrases from various sources. However, for a long document, the resulting list of candidates can be long. Consequently, different *pruning* heuristics have been designed to prune candidates that are unlikely to be keyphrases (Huang et al., 2006; Kumar and Srinathan, 2008; El-Beltagy and Rafea, 2009; You et al., 2009; Newman et al., 2012).

3.2 Supervised Approaches

Research on supervised approaches to keyphrase extraction has focused on two issues: *task reformulation* and *feature design*.

3.2.1 Task Reformulation

Early supervised approaches to keyphrase extraction recast this task as a binary *classification* problem (Frank et al., 1999; Turney, 1999; Witten et al., 1999; Turney, 2000). The goal is to train a classifier on documents annotated with keyphrases to determine whether a candidate phrase is a keyphrase. Keyphrases and non-keyphrases are used to generate positive and negative examples, respectively. Different learning algorithms have been used to train this classifier, including naïve Bayes (Frank et al., 1999; Witten et al., 1999), decision trees (Turney, 1999; Turney, 2000), bagging (Hulth, 2003), boosting (Hulth et al., 2001), maximum entropy (Yih et al., 2006; Kim and Kan, 2009), multi-layer perceptron (Lopez and Romary, 2010), and support vector machines (Jiang et al., 2009; Lopez and Romary, 2010).

Recasting keyphrase extraction as a classification problem has its weaknesses, however. Recall that the goal of keyphrase extraction is to identify the most representative phrases for a document. In other words, if a candidate phrase c_1 is more representative than another candidate phrase c_2, c_1 should be preferred to c_2. Note that a binary classifier classifies each candidate keyphrase independently of the others, and consequently it does not allow us to determine which candidates are better than the others (Hulth, 2004; Wang and Li, 2011).

Motivated by this observation, Jiang et al. (2009) propose a *ranking* approach to keyphrase extraction, where the goal is to learn a ranker to rank two candidate keyphrases. This pairwise ranking approach therefore introduces competition between candidate keyphrases, and has been shown to significantly outperform KEA (Witten et al., 1999; Frank et al., 1999), a popular supervised baseline that adopts the traditional supervised classification approach (Song et al., 2003; Kelleher and Luz, 2005).

3.2.2 Features

The features commonly used to represent an instance for supervised keyphrase extraction can be broadly divided into two categories.

3.2.2.1 Within-Collection Features

Within-collection features are computed based solely on the training documents. These features can be further divided into three types.

Statistical features are computed based on statistical information gathered from the training documents. Three such features have been extensively used in supervised approaches. The first one, *tf*idf* (Salton and Buckley, 1988), is computed based on candidate frequency in the given text and inverse document frequency (i.e., number of other documents where the candidate appears).[2] The second one, the *distance* of a phrase, is defined as the number of words preceding its first occurrence normalized by the number of words in the document. Its usefulness stems from the fact that keyphrases tend to appear early in a document. The third one, *supervised keyphraseness*, encodes the number of times a phrase appears as a keyphrase in the training set. This feature is designed based on the assumption that a phrase frequently tagged as a keyphrase is more likely to be a keyphrase in an unseen document. These three features form the feature set of KEA (Witten et al., 1999; Frank et al., 1999), and have been shown to perform consistently well on documents from various sources (Yih et al., 2006; Kim et al., 2013). Other statistical features include *phrase length* and *spread* (i.e., the number of words between the first and last occurrences of a phrase in the document).

Structural features encode how different instances of a candidate keyphrase are located in different parts of a document. A phrase is more likely to be a keyphrase if it appears in the abstract or introduction of a paper or in the metadata section of a web page. In fact, features that encode how frequently a candidate keyphrase occurs in various sections of a scientific paper (e.g., introduction, conclusion) (Nguyen and Kan, 2007) and those that encode the location of a candidate keyphrase in a web page (e.g., whether it appears in the title) (Chen et al., 2005; Yih et al., 2006) have been shown to be useful for the task.

Syntactic features encode the syntactic patterns of a candidate keyphrase. For example, a candidate keyphrase has been encoded as (1) a *PoS tag sequence*, which denotes the sequence of part-of-speech tag(s) assigned to its word(s); and (2) a *suffix sequence*, which is the sequence of morphological suffixes of its words (Yih et al., 2006; Nguyen and Kan, 2007; Kim and Kan, 2009). However, ablation studies conducted on web pages (Yih et al., 2006) and scientific articles

[2] A tf*idf-based baseline, where candidate keyphrases are ranked and selected according to tf*idf, has been widely used by both supervised and unsupervised approaches (Zhang et al., 2005; Dredze et al., 2008; Paukkeri et al., 2008; Grineva et al., 2009).

(Kim and Kan, 2009) reveal that syntactic features are not useful for keyphrase extraction in the presence of other feature types.

3.2.2.2 External Resource-Based Features

External resource-based features are computed based on information gathered from resources other than the training documents, such as lexical knowledge bases (e.g., Wikipedia) or the Web, with the goal of improving keyphrase extraction performance by exploiting external knowledge. Below we give an overview of the external resource-based features that have proven useful for keyphrase extraction.

Wikipedia-based keyphraseness is computed as a candidate's document frequency multiplied by the ratio of the number of Wikipedia articles where the candidate appears as a link to the number of articles where it appears (Medelyan et al., 2009). This feature is motivated by the observation that a candidate is likely to be a keyphrase if it occurs frequently as a link in Wikipedia. Unlike supervised keyphraseness, Wikipedia-based keyphraseness can be computed without using documents annotated with keyphrases and can work even if there is a mismatch between the training domain and the test domain.

Yih et al. (2006) employ a feature that encodes whether a candidate keyphrase appears in the *query log* of a search engine, exploiting the observation that a candidate is potentially important if it was used as a search query. Terminological databases have been similarly exploited to encode the salience of candidate keyphrases in scientific papers (Lopez and Romary, 2010).

While the aforementioned external resource-based features attempt to encode how salient a candidate keyphrase is, Turney (2003) proposes features that encode the semantic relatedness between two candidate keyphrases. Noting that candidate keyphrases that are not semantically related to the predicted keyphrases are unlikely to be keyphrases in technical reports, Turney employs *coherence features* to identify such candidate keyphrases. Semantic relatedness is encoded in the coherence features as two candidate keyphrases' pointwise mutual information, which Turney computes by using the Web as a corpus.

3.3 Unsupervised Approaches

Existing unsupervised approaches to keyphrase extraction can be categorized into four groups.

3.3.1 Graph-Based Ranking

Intuitively, keyphrase extraction is about finding the important words and phrases from a document. Traditionally, the *importance* of a candidate has often been defined in terms of how related it is to other candidates in the document. Informally, a candidate is important if it is related to (1) a large number of candidates and (2) candidates that are important. Researchers have computed *relatedness* between candidates using co-occurrence counts (Mihalcea and Tarau, 2004; Matsuo and Ishizuka, 2004) and semantic relatedness (Grineva et al., 2009), and represented the relatedness information collected from a document as a graph (Mihalcea and Tarau, 2004; Wan and Xiao, 2008a; Wan and Xiao, 2008b; Bougouin et al., 2013).

The basic idea behind a graph-based approach is to build a graph from the input document and rank its nodes according to their importance using a graph-based ranking method (e.g., Brin and Page (1998)). Each node of the graph corresponds to a candidate keyphrase from the document and an edge connects two *related* candidates. The edge weight is proportional to the syntactic and/or semantic relevance between the connected candidates. For each node, each of its edges is treated as a "vote" from the other node connected by the edge. A node's score in the graph is defined recursively in terms of the edges it has and the scores of the neighboring nodes. The top-ranked candidates from the graph are then selected as keyphrases for the input document. TextRank (Mihalcea and Tarau, 2004) is one of the most well-known graph-based approaches to keyphrase extraction.

This instantiation of a graph-based approach overlooks an important aspect of keyphrase extraction, however. A set of keyphrases for a document should ideally cover the main topics discussed in it, but this instantiation does not guarantee that all the main topics will be represented by the extracted keyphrases. Despite this weakness, a graph-based representation of text was adopted by many approaches that propose different ways of computing the similarity between two candidates.

3.3.2 Topic-Based Clustering

Another unsupervised approach to keyphrase extraction involves grouping the candidate keyphrases in a document into *topics*, such that each topic is composed of all and only those candidate keyphrases that are related to that topic (Grineva et al., 2009; Liu et al., 2009b; Liu et

al., 2010). There are several motivations behind this topic-based clustering approach. First, a keyphrase should ideally be relevant to one or more main topic(s) discussed in a document (Liu et al., 2010; Liu et al., 2012). Second, the extracted keyphrases should be comprehensive in the sense that they should cover all the main topics in a document (Liu et al., 2009b; Liu et al., 2010; Liu et al., 2012). Below we examine three representative systems that adopt this approach.

KeyCluster Liu et al. (2009b) adopt a clustering-based approach (henceforth KeyCluster) that clusters semantically similar candidates using Wikipedia and co-occurrence-based statistics. The underlying hypothesis is that each of these clusters corresponds to a topic covered in the document, and selecting the candidates close to the centroid of each cluster as keyphrases ensures that the resulting set of keyphrases covers all the topics of the document.

While empirical results show that KeyCluster performs better than both TextRank and Hulth's (2003) supervised system, KeyCluster has a potential drawback: by extracting keyphrases from each topic cluster, it essentially gives each topic equal importance. In practice, however, there could be topics that are not important and these topics should not have keyphrase(s) representing them.

Topical PageRank (TPR) Liu et al. (2010) propose TPR, an approach that overcomes the aforementioned weakness of KeyCluster. It runs TextRank multiple times for a document, once for each of its topics induced by a Latent Dirichlet Allocation (Blei et al., 2003). By running TextRank once for each topic, TPR ensures that the extracted keyphrases cover the main topics of the document. The final score of a candidate is computed as the sum of its scores for each of the topics, weighted by the probability of that topic in that document. Hence, unlike KeyCluster, candidates belonging to a less probable topic are given less importance.

TPR performs significantly better than both tf*idf and TextRank on the DUC-2001 and *Inspec* datasets. TPR's superior performance strengthens the hypothesis of using topic clustering for keyphrase extraction. However, though TPR is conceptually better than KeyCluster, Liu et al. did not compare TPR against KeyCluster.

CommunityCluster Grineva et al. (2009) propose CommunityCluster, a variant of the topic clustering approach to keyphrase extraction. Like TPR, CommunityCluster gives more weight to more important topics, but unlike TPR, it extracts *all* candidate keyphrases from an important topic, assuming that a candidate that receives little focus in the text should still be extracted as a keyphrase as long as it is related to an important topic. CommunityCluster yields much better recall (without losing precision) than extractors such as tf*idf, TextRank, and the Yahoo! term extractor.

3.3.3 Simultaneous Learning

Since keyphrases represent a dense summary of a document, researchers hypothesized that text summarization and keyphrase extraction can potentially benefit from each other if these tasks are performed simultaneously. Zha (2002) proposes the first graph-based approach for simultaneous summarization and keyphrase extraction, motivated by a key observation: a sentence is important if it contains important words, and important words appear in important sentences. Wan et al. (2007) extend Zha's work by adding two assumptions: (1) an important sentence is connected to other important sentences, and (2) an important word is linked to other important words, a TextRank-like assumption. Based on these assumptions, Wan et al. (2007) build three graphs to capture the association between the sentences (S) and the words (W) in an input document, namely, a S–S graph, a bipartite S–W graph, and a W–W graph. The weight of an edge connecting two sentence nodes in a S–S graph corresponds to their content similarity. An edge weight in a S–W graph denotes the word's importance in the sentence it appears. Finally, an edge weight in a W–W graph denotes the co-occurrence or knowledge-based similarity between the two connected words. Once the graphs are constructed for an input document, an iterative reinforcement algorithm is applied to assign scores to each sentence and word. The top-scored words are used to form keyphrases.

The main advantage of this approach is that it combines the strengths of both Zha's approach (i.e., bipartite S–W graphs) and TextRank (i.e., W–W graphs) and performs better than both of them. However, it has a weakness: like TextRank, it does not ensure that the extracted keyphrases will cover all the main topics. To address this problem, one can employ a topic clustering algorithm on the W–W graph to produce the topic clusters, and then ensure that keyphrases are chosen from every main topic cluster.

3.3.4 Language Modeling

Many existing approaches have a separate, heuristic module for extracting candidate keyphrases prior to keyphrase ranking/extraction. In contrast, Tomokiyo and Hurst (2003) propose an approach (henceforth LMA) that combines these two steps.

LMA scores a candidate keyphrase based on two features, namely, *phraseness* (i.e., the extent to which a word sequence can be treated as a phrase) and *informativeness* (i.e., the extent to which a word sequence captures the central idea of the document it appears in). Intuitively, a phrase that has high scores for phraseness and informativeness is likely to be a keyphrase. These feature values are estimated using language models (LMs) trained on a *foreground* corpus and a *background* corpus. The foreground corpus is composed of the set of documents from which keyphrases are to be extracted. The background corpus is a large corpus that encodes general knowledge about the world (e.g., the Web). A unigram LM and an n-gram LM are constructed for each of these two corpora. Phraseness, defined using the foreground LM, is calculated as the loss of information incurred as a result of assuming a unigram LM (i.e., conditional independence among the words of the phrase) instead of an n-gram LM (i.e., the phrase is drawn from an n-gram LM). Informativeness is computed as the loss that results because of the assumption that the candidate is sampled from the background LM rather than the foreground LM. The loss values are computed using Kullback-Leibler divergence. Candidates are ranked according to the sum of these two feature values.

In sum, LMA uses a language model rather than heuristics to identify phrases, and relies on the language model trained on the background corpus to determine how "unique" a candidate keyphrase is to the domain represented by the foreground corpus. The more unique it is to the foreground's domain, the more likely it is a keyphrase for that domain. While the use of language models to identify phrases cannot be considered a major strength of this approach (because heuristics can identify phrases fairly reliably), the use of a background corpus to identify candidates that are unique to the foreground's domain is a unique aspect of this approach. We believe that this idea deserves further investigation, as it would allow us to discover a keyphrase that is unique to the foreground's domain but may have a low tf*idf value.

4 Evaluation

In this section, we describe metrics for evaluating keyphrase extraction systems as well as state-of-the-art results on commonly-used datasets.

4.1 Evaluation Metrics

Designing evaluation metrics for keyphrase extraction is by no means an easy task. To score the output of a keyphrase extraction system, the typical approach, which is also adopted by the SemEval-2010 shared task on keyphrase extraction, is (1) to create a mapping between the keyphrases in the gold standard and those in the system output using *exact match*, and then (2) score the output using evaluation metrics such as precision (P), recall (R), and F-score (F).

Conceivably, exact match is an overly strict condition, considering a predicted keyphrase incorrect even if it is a variant of a gold keyphrase. For instance, given the gold keyphrase "neural network", exact match will consider a predicted phrase incorrect even if it is an expanded version of the gold keyphrase ("artificial neural network") or one of its morphological ("neural networks") or lexical ("neural net") variants. While morphological variations can be handled using a stemmer (El-Beltagy and Rafea, 2009), other variations may not be handled easily and reliably.

Human evaluation has been suggested as a possibility (Matsuo and Ishizuka, 2004), but it is time-consuming and expensive. For this reason, researchers have experimented with two types of automatic evaluation metrics. The first type of metrics addresses the problem with exact match. These metrics reward a partial match between a predicted keyphrase and a gold keyphrase (i.e., overlapping n-grams) and are commonly used in machine translation (MT) and summarization evaluations. They include BLEU, METEOR, NIST, and ROUGE. Nevertheless, experiments show that these MT metrics only offer a partial solution to problem with exact match: they can only detect a subset of the near-misses (Kim et al., 2010a).

The second type of metrics focuses on how a system ranks its predictions. Given that two systems A and B have the same number of correct predictions, binary preference measure (Bpref) and mean reciprocal rank (MRR) (Liu et al., 2010) will award more credit to A than to B if the ranks of the correct predictions in A's output are higher than those in B's output. R-precision (R_p) is an

IR metric that focuses on ranking: given a document with n gold keyphrases, it computes the precision of a system over its n highest-ranked candidates (Zesch and Gurevych, 2009). The motivation behind the design of R_p is simple: a system will achieve a perfect R_p value if it ranks all the keyphrases above the non-keyphrases.

4.2 The State of the Art

Table 2 lists the best scores on some popular evaluation datasets and the corresponding systems. For example, the best F-scores on the *Inspec* test set, the DUC-2001 dataset, and the SemEval-2010 test set are 45.7, 31.7, and 27.5, respectively.[3]

Two points deserve mention. First, F-scores decrease as document length increases. These results are consistent with the observation we made in Section 2 that it is more difficult to extract keyphrases correctly from longer documents. Second, recent unsupervised approaches have rivaled their supervised counterparts in performance (Mihalcea and Tarau, 2004; El-Beltagy and Rafea, 2009; Liu et al., 2009b). For example, KP-Miner (El-Beltagy and Rafea, 2010), an unsupervised system, ranked third in the SemEval-2010 shared task with an F-score of 25.2, which is comparable to the best supervised system scoring 27.5.

5 Analysis

With the goal of providing directions for future work, we identify the errors commonly made by state-of-the-art keyphrase extractors below.

5.1 Error Analysis

Although a few researchers have presented a sample of their systems' output and the corresponding gold keyphrases to show the differences between them (Witten et al., 1999; Nguyen and Kan, 2007; Medelyan et al., 2009), a systematic analysis of the major types of errors made by state-of-the-art keyphrase extraction systems is missing.

To fill this gap, we ran four keyphrase extraction systems on four commonly-used datasets of varying sources, including *Inspec* abstracts (Hulth, 2003), DUC-2001 news articles (Over, 2001), scientific papers (Kim et al., 2010b), and meeting transcripts (Liu et al., 2009a). Specifically, we randomly selected 25 documents from each of these four datasets and manually analyzed the output of the four systems, including tf*idf, the most frequently used baseline, as well as three state-of-the-art keyphrase extractors, of which two are unsupervised (Wan and Xiao, 2008b; Liu et al., 2009b) and one is supervised (Medelyan et al., 2009).

Dataset	Approach and System [Supervised?]	Score		
		P	R	F
Abstracts (*Inspec*)	Topic clustering (Liu et al., 2009b) [×]	35.0	66.0	45.7
Blogs	Topic community detection (Grineva et al., 2009) [×]	35.1	61.5	44.7
News (DUC -2001)	Graph-based ranking for extended neighborhood (Wan and Xiao, 2008b) [×]	28.8	35.4	31.7
Papers (SemEval -2010)	Statistical, semantic, and distributional features (Lopez and Romary, 2010) [✓]	27.2	27.8	27.5

Table 2: Best scores achieved on various datasets.

Our analysis reveals that the errors fall into four major types, each of which contributes significantly to the overall errors made by the four systems, despite the fact that the contribution of each of these error types varies from system to system. Moreover, we do not observe any significant difference between the types of errors made by the four systems other than the fact that the supervised system has the expected tendency to predict keyphrases seen in the training data. Below we describe these four major types of errors.

Overgeneration errors are a major type of precision error, contributing to 28–37% of the overall error. Overgeneration errors occur when a system correctly predicts a candidate as a keyphrase because it contains a word that appears frequently in the associated document, but at the same time erroneously outputs other candidates as keyphrases because they contain the same word. Recall that for many systems, it is not easy to reject a non-keyphrase containing a word with a high term frequency: many unsupervised systems score a candidate by summing the score of each of its component words, and many supervised systems use unigrams as features to represent a candidate. To be more concrete, consider the news article on athlete *Ben Johnson* in Figure 1, where the keyphrases are boldfaced. As we can see, the word *Olympic(s)* has a significant presence in the document. Consequently, many systems not only correctly predict *Olympics* as a keyphrase, but also erroneously predict *Olympic movement* as a keyphrase, yielding overgeneration errors.

Infrequency errors are a major type of re-

[3] A more detailed analysis of the results of the SemEval-2010 shared task and the approaches adopted by the participating systems can be found in Kim et al. (2013).

Canadian **Ben Johnson** left the **Olympics** today "in a complete state of shock," accused of cheating with drugs in the world's fastest **100-meter dash** and stripped of his **gold medal**. The prize went to American **Carl Lewis**. Many athletes accepted the accusation that Johnson used a muscle-building but dangerous and illegal anabolic steroid called **stanozolol** as confirmation of what they said they know has been going on in track and field. Two tests of Johnson's urine sample proved positive and his denials of **drug use** were rejected today. "This is a blow for the Olympic Games and the Olympic movement," said International Olympic Committee President Juan Antonio Samaranch.

Figure 1: A news article on *Ben Johnson* from the DUC-2001 dataset. The keyphrases are boldfaced.

call error contributing to 24–27% of the overall error. Infrequency errors occur when a system fails to identify a keyphrase owing to its infrequent presence in the associated document (Liu et al., 2011). Handling infrequency errors is a challenge because state-of-the-art keyphrase extractors rarely predict candidates that appear only once or twice in a document. In the *Ben Johnson* example, many keyphrase extractors fail to identify *100-meter dash* and *gold medal* as keyphrases, resulting in infrequency errors.

Redundancy errors are a type of precision error contributing to 8–12% of the overall error. Redundancy errors occur when a system correctly identifies a candidate as a keyphrase, but at the same time outputs a semantically equivalent candidate (e.g., its alias) as a keyphrase. This type of error can be attributed to a system's failure to determine that two candidates are semantically equivalent. Nevertheless, some researchers may argue that a system should not be penalized for redundancy errors because the extracted candidates are in fact keyphrases. In our example, *Olympics* and *Olympic games* refer to the same concept, so a system that predicts both of them as keyphrases commits a redundancy error.

Evaluation errors are a type of recall error contributing to 7–10% of the overall error. An evaluation error occurs when a system outputs a candidate that is semantically equivalent to a gold keyphrase, but is considered erroneous by a scoring program because of its failure to recognize that the predicted phrase and the corresponding gold keyphrase are semantically equivalent. In other words, an evaluation error is not an error made by a keyphrase extractor, but an error due to the naivety of a scoring program. In our example, while *Olympics* and *Olympic games* refer to the same concept, only the former is annotated as keyphrase. Hence, an evaluation error occurs if a system predicts *Olympic games* but not *Olympics* as a keyphrase and the scoring program fails to identify them as semantically equivalent.

5.2 Recommendations

We recommend that *background knowledge* be extracted from external lexical databases (e.g., YAGO2 (Suchanek et al., 2007), Freebase (Bollacker et al., 2008), BabelNet (Navigli and Ponzetto, 2012)) to address the four types of errors discussed above.

First, we discuss how **redundancy errors** could be addressed by using the background knowledge extracted from external databases. Note that if we can identify semantically equivalent candidates, then we can reduce redundancy errors. The question, then, is: can background knowledge be used to help us identify semantically equivalent candidates? To answer this question, note that Freebase, for instance, has over 40 million *topics* (i.e., real-world entities such as people, places, and things) from over 70 domains (e.g., music, business, education). Hence, before a system outputs a set of candidates as keyphrases, it can use Freebase to determine whether any of them is mapped to the same Freebase topic. Referring back to our running example, both *Olympics* and *Olympic games* are mapped to a Freebase topic called *Olympic games*. Based on this information, a keyphrase extractor can determine that the two candidates are aliases and should output only one of them, thus preventing a redundancy error.

Next, we discuss how **infrequency errors** could be addressed using background knowledge. A natural way to handle this problem would be to make an infrequent keyphrase frequent. To accomplish this, we suggest exploiting an influential idea in the keyphrase extraction literature: the importance of a candidate is defined in terms of how related it is to other candidates in the text (see Section 3.3.1). In other words, if we could relate an infrequent keyphrase to other candidates in the text, we could boost its importance.

We believe that this could be accomplished using background knowledge. The idea is to boost the importance of infrequent keyphrases using their frequent counterparts. Consider again our running example. All four systems have managed to identify *Ben Johnson* as a keyphrase due to its

significant presence. Hence, we can boost the importance of *100-meter dash* and *gold medal* if we can relate them to *Ben Johnson*.

To do so, note that Freebase maps a candidate to one or more pre-defined topics, each of which is associated with one or more types. Types are similar to entity classes. For instance, the candidate *Ben Johnson* is mapped to a Freebase topic with the same name, which is associated with Freebase types such as *Person*, *Athlete*, and *Olympic athlete*. Types are defined for a specific domain in Freebase. For instance, *Person*, *Athlete*, and *Olympic athlete* are defined in the *People*, *Sports*, and *Olympics* domains, respectively. Next, consider the two infrequent candidates, *100-meter dash* and *gold medal*. *100-meter dash* is mapped to the topic *Sprint* of type *Sports* in the *Sports* domain, whereas *gold medal* is mapped to a topic with the same name of type *Olympic medal* in the *Olympics* domain. Consequently, we can relate *100-meter dash* to *Ben Johnson* via the *Sports* domain (i.e., they belong to different types under the same domain). Additionally, *gold medal* can be related to *Ben Johnson* via the *Olympics* domain.

As discussed before, the relationship between two candidates is traditionally established using co-occurrence information. However, using co-occurrence windows has its shortcomings. First, an *ad-hoc* window size cannot capture related candidates that are not inside the window. So it is difficult to predict *100-meter dash* and *gold medal* as keyphrases: they are more than 10 tokens away from frequent words such as *Johnson* and *Olympics*. Second, the candidates inside a window are all assumed to be related to each other, but it is apparently an overly simplistic assumption. There have been a few attempts to design Wikipedia-based relatedness measures, with promising initial results (Grineva et al., 2009; Liu et al., 2009b; Medelyan et al., 2009).[4]

Overgeneration errors could similarly be addressed using background knowledge. Recall that *Olympic movement* is not a keyphrase in our example although it includes an important word (i.e., *Olympic*). Freebase maps *Olympic movement* to a topic with the same name, which is associated with a type called *Musical Recording* in the *Music* domain. However, it does not map *Olympic movement* to any topic in the *Olympics* domain. The absence of such a mapping in the *Olympics* domain could be used by a keyphrase extractor as a supporting evidence against predicting *Olympic movement* as a keyphrase.

Finally, as mentioned before, **evaluation errors** should not be considered errors made by a system. Nevertheless, they reveal a problem with the way keyphrase extractors are currently evaluated. To address this problem, one possibility is to conduct human evaluations. Cheaper alternatives include having human annotators identify semantically equivalent keyphrases during manual labeling, and designing scoring programs that can automatically identify such semantic equivalences.

6 Conclusion and Future Directions

We have presented a survey of the state of the art in automatic keyphrase extraction. While unsupervised approaches have started to rival their supervised counterparts in performance, the task is far from being solved, as reflected by the fairly poor state-of-the-art results on various commonly-used evaluation datasets. Our analysis revealed that there are at least three major challenges ahead.

1. Incorporating background knowledge. While much recent work has focused on algorithmic development, keyphrase extractors need to have a deeper "understanding" of a document in order to reach the next level of performance. Such an understanding can be facilitated by the incorporation of background knowledge.

2. Handling long documents. While it may be possible to design better algorithms to handle the large number of candidates in long documents, we believe that employing sophisticated features, especially those that encode background knowledge, will enable keyphrases and non-keyphrases to be distinguished more easily even in the presence of a large number of candidates.

3. Improving evaluation schemes. To more accurately measure the performance of keyphrase extractors, they should not be penalized for evaluation errors. We have suggested several possibilities as to how this problem can be addressed.

Acknowledgments

We thank the anonymous reviewers for their detailed and insightful comments on earlier drafts of this paper. This work was supported in part by NSF Grants IIS-1147644 and IIS-1219142.

[4]Note that it may be difficult to employ our recommendations to address infrequency errors in informal text with uncorrelated topics because the keyphrases it contains may not be related to each other (see Section 2).

References

Ken Barker and Nadia Cornacchia. 2000. Using noun phrase heads to extract document keyphrases. In *Proceedings of the 13th Biennial Conference of the Canadian Society on Computational Studies of Intelligence*, pages 40–52.

Gábor Berend. 2011. Opinion expression mining by exploiting keyphrase extraction. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*, pages 1162–1170.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250.

Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. Topicrank: Graph-based topic ranking for keyphrase extraction. In *Proceedings of the 6th International Joint Conference on Natural Language Processing*, pages 543–551.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks*, 30(1–7):107–117.

Mo Chen, Jian-Tao Sun, Hua-Jun Zeng, and Kwok-Yan Lam. 2005. A practical system of keyphrase extraction for web pages. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pages 277–278.

Zhuoye Ding, Qi Zhang, and Xuanjing Huang. 2011. Keyphrase extraction from online news using binary integer programming. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*, pages 165–173.

Mark Dredze, Hanna M. Wallach, Danny Puller, and Fernando Pereira. 2008. Generating summary keywords for emails using topics. In *Proceedings of the 13th International Conference on Intelligent User Interfaces*, pages 199–206.

Samhaa R. El-Beltagy and Ahmed A. Rafea. 2009. KP-Miner: A keyphrase extraction system for English and Arabic documents. *Information Systems*, 34(1):132–144.

Samhaa R. El-Beltagy and Ahmed Rafea. 2010. KP-Miner: Participation in SemEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 190–193.

Eibe Frank, Gordon W. Paynter, Ian H. Witten, Carl Gutwin, and Craig G. Nevill-Manning. 1999. Domain-specific keyphrase extraction. In *Proceedings of 16th International Joint Conference on Artificial Intelligence*, pages 668–673.

Maria Grineva, Maxim Grinev, and Dmitry Lizorkin. 2009. Extracting key terms from noisy and multitheme documents. In *Proceedings of the 18th International Conference on World Wide Web*, pages 661–670.

Carl Gutwin, Gordon Paynter, Ian Witten, Craig Nevill-Manning, and Eibe Frank. 1999. Improving browsing in digital libraries with keyphrase indexes. *Decision Support Systems*, 27:81–104.

Khaled M. Hammouda, Diego N. Matute, and Mohamed S. Kamel. 2005. CorePhrase: Keyphrase extraction for document clustering. In *Proceedings of the 4th International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 265–274.

Kazi Saidul Hasan and Vincent Ng. 2010. Conundrums in unsupervised keyphrase extraction: Making sense of the state-of-the-art. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 365–373.

Chong Huang, Yonghong Tian, Zhi Zhou, Charles X. Ling, and Tiejun Huang. 2006. Keyphrase extraction using semantic networks structure analysis. In *Proceedings of the 6th International Conference on Data Mining*, pages 275–284.

Anette Hulth and Beáta B. Megyesi. 2006. A study on automatically extracted keywords in text categorization. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 537–544.

Anette Hulth, Jussi Karlgren, Anna Jonsson, Henrik Boström, and Lars Asker. 2001. Automatic keyword extraction using domain knowledge. In *Proceedings of the 2nd International Conference on Computational Linguistics and Intelligent Text Processing*, pages 472–482.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 216–223.

Anette Hulth. 2004. Enhancing linguistically oriented automatic keyword extraction. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: Short Papers*, pages 17–20.

Xin Jiang, Yunhua Hu, and Hang Li. 2009. A ranking approach to keyphrase extraction. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 756–757.

Daniel Kelleher and Saturnino Luz. 2005. Automatic hypertext keyphrase detection. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 1608–1609.

Su Nam Kim and Timothy Baldwin. 2012. Extracting keywords from multi-party live chats. In *Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation*, pages 199–208.

Su Nam Kim and Min-Yen Kan. 2009. Re-examining automatic keyphrase extraction approaches in scientific articles. In *Proceedings of the ACL-IJCNLP Workshop on Multiword Expressions*, pages 9–16.

Su Nam Kim, Timothy Baldwin, and Min-Yen Kan. 2010a. Evaluating n-gram based evaluation metrics for automatic keyphrase extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 572–580.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010b. SemEval-2010 Task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 21–26.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2013. Automatic keyphrase extraction from scientific articles. *Language Resources and Evaluation*, 47(3):723–742.

Niraj Kumar and Kannan Srinathan. 2008. Automatic keyphrase extraction from scientific documents using n-gram filtration technique. In *Proceedings of the 8th ACM Symposium on Document Engineering*, pages 199–208.

Feifan Liu, Deana Pennell, Fei Liu, and Yang Liu. 2009a. Unsupervised approaches for automatic keyword extraction using meeting transcripts. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 620–628.

Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. 2009b. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 257–266.

Zhiyuan Liu, Wenyi Huang, Yabin Zheng, and Maosong Sun. 2010. Automatic keyphrase extraction via topic decomposition. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 366–376.

Zhiyuan Liu, Xinxiong Chen, Yabin Zheng, and Maosong Sun. 2011. Automatic keyphrase extraction by bridging vocabulary gap. In *Proceedings of the 15th Conference on Computational Natural Language Learning*, pages 135–144.

Zhiyuan Liu, Chen Liang, and Maosong Sun. 2012. Topical word trigger model for keyphrase extraction. In *Proceedings of the 24th International Conference on Computational Linguistics*, pages 1715–1730.

Patrice Lopez and Laurent Romary. 2010. HUMB: Automatic key term extraction from scientific articles in GROBID. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 248–251.

Yutaka Matsuo and Mitsuru Ishizuka. 2004. Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools*, 13.

Olena Medelyan, Eibe Frank, and Ian H. Witten. 2009. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1318–1327.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into texts. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411.

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

David Newman, Nagendra Koilada, Jey Han Lau, and Timothy Baldwin. 2012. Bayesian text segmentation for index term identification and keyphrase extraction. In *Proceedings of the 24th International Conference on Computational Linguistics*, pages 2077–2092.

Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *Proceedings of the International Conference on Asian Digital Libraries*, pages 317–326.

Chau Q. Nguyen and Tuoi T. Phan. 2009. An ontology-based approach for key phrase extraction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing: Short Papers*, pages 181–184.

Paul Over. 2001. Introduction to DUC-2001: An intrinsic evaluation of generic news text summarization systems. In *Proceedings of the 2001 Document Understanding Conference*.

Mari-Sanna Paukkeri, Ilari T. Nieminen, Matti Pöllä, and Timo Honkela. 2008. A language-independent approach to keyphrase extraction and evaluation. In *Proceedings of the 22nd International Conference on Computational Linguistics: Companion Volume: Posters*, pages 83–86.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523.

Min Song, Il-Yeol Song, and Xiaohua Hu. 2003. KPSpotter: A flexible information gain-based keyphrase extraction system. In *Proceedings of the 5th ACM International Workshop on Web Information and Data Management*, pages 50–53.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. YAGO: A core of semantic knowledge. In *Proceedings of the 16th International World Wide Web Conference*, pages 697–706.

Takashi Tomokiyo and Matthew Hurst. 2003. A language model approach to keyphrase extraction. In *Proceedings of the ACL Workshop on Multiword Expressions*, pages 33–40.

Peter Turney. 1999. Learning to extract keyphrases from text. *National Research Council Canada, Institute for Information Technology, Technical Report ERB-1057*.

Peter Turney. 2000. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2:303–336.

Peter Turney. 2003. Coherent keyphrase extraction via web mining. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 434–439.

Xiaojun Wan and Jianguo Xiao. 2008a. CollabRank: Towards a collaborative approach to single-document keyphrase extraction. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 969–976.

Xiaojun Wan and Jianguo Xiao. 2008b. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 855–860.

Xiaojun Wan, Jianwu Yang, and Jianguo Xiao. 2007. Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 552–559.

Chen Wang and Sujian Li. 2011. CoRankBayes: Bayesian learning to rank under the co-training framework and its application in keyphrase extraction. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pages 2241–2244.

Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill-Manning. 1999. KEA: Practical automatic keyphrase extraction. In *Proceedings of the 4th ACM Conference on Digital Libraries*, pages 254–255.

Yi-Fang Brook Wu, Quanzhi Li, Razvan Stefan Bot, and Xin Chen. 2005. Domain-specific keyphrase extraction. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pages 283–284.

Wen-Tau Yih, Joshua Goodman, and Vitor R. Carvalho. 2006. Finding advertising keywords on web pages. In *Proceedings of the 15th International Conference on World Wide Web*, pages 213–222.

Wei You, Dominique Fontaine, and Jean-Paul Barthès. 2009. Automatic keyphrase extraction with a refined candidate set. In *Proceedings of the IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, pages 576–579.

Torsten Zesch and Iryna Gurevych. 2009. Approximate matching for evaluating keyphrase extraction. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing 2009*, pages 484–489.

Hongyuan Zha. 2002. Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering. In *Proceedings of 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 113–120.

Yongzheng Zhang, Nur Zincir-Heywood, and Evangelos Milios. 2004. World Wide Web site summarization. *Web Intelligence and Agent Systems*, 2:39–53.

Yongzheng Zhang, Nur Zincir-Heywood, and Evangelos Milios. 2005. Narrative text classification for automatic key phrase extraction in web document corpora. In *Proceedings of the 7th ACM International Workshop on Web Information and Data Management*, pages 51–58.

Xin Zhao, Jing Jiang, Jing He, Yang Song, Palakorn Achanauparp, Ee-Peng Lim, and Xiaoming Li. 2011. Topical keyphrase extraction from Twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 379–388.

Pattern Dictionary of English Prepositions

Ken Litkowski
CL Research
9208 Gue Road
Damascus, MD 20872 USA
ken@clres.com

Abstract

We present a new lexical resource for the study of preposition behavior, the Pattern Dictionary of English Prepositions (PDEP). This dictionary, which follows principles laid out in Hanks' theory of norms and exploitations, is linked to 81,509 sentences for 304 prepositions, which have been made available under The Preposition Project (TPP). Notably, 47,285 sentences, initially untagged, provide a representative sample of preposition use, unlike the tagged sentences used in previous studies. Each sentence has been parsed with a dependency parser and our system has near-instantaneous access to features developed with this parser to explore and annotate properties of individual senses. The features make extensive use of WordNet. We have extended feature exploration to include lookup of FrameNet lexical units and VerbNet classes for use in characterizing preposition behavior. We have designed our system to allow public access to any of the data available in the system.

1 Introduction

Recent studies (Zapirain et al. (2013); Srikumar and Roth (2011)) have shown the value of prepositional phrases in joint modeling with verbs for semantic role labeling. Although recent studies have shown improved preposition disambiguation, they have received little systematic treatment from a lexicographic perspective. Recently, a new corpus has been made available that promises to be much more representative of preposition behavior. Our initial examination of this corpus has suggested clear indications of senses previously overlooked and reduced prominence for senses thought to constitute a large role in preposition use.

In section 2, we describe the interface to the Pattern Dictionary of English Prepositions (PDEP), identifying how we are building upon data developed in The Preposition Project (TPP) and investigating its sense inventory with corpora also made available under TPP. Section 3 describes the procedures for tagging a representative corpus drawn from the British National Corpus, including some findings that have emerged in assessing previous studies of preposition disambiguation. Section 4 describes how we are able to investigate the relationship of WordNet, FrameNet, and VerbNet to this effort and how this examination of preposition behavior can be used in working with these resources. Section 5 describes how we can use PDEP for the analysis of semantic role and semantic relation inventories. Section 6 describes how we envision further developments of PDEP and how the data are available for further analysis. In section 7, we present our conclusions for PDEP.

2 The Pattern Dictionary of English Prepositions

Litkowski and Hargraves (2005) and Litkowski and Hargraves (2006) describe The Preposition Project (TPP) as an attempt to describe preposition behavior using a sense inventory made available for public use from the *Oxford Dictionary of English* (Stevenson and Soanes, 2003) by tagging sentences drawn from FrameNet. In TPP, each sense was characterized with its complement and attachment (or governor) properties, its class and semantic relation, substitutable prepositions, its syntactic positions, and any FrameNet frame and frame element usages (where available). The FrameNet sentences were sense-tagged using the sense inventory and were

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1274–1283,
Baltimore, Maryland, USA, June 23-25 2014.

later used as the basis for a preposition disambiguation task in SemEval 2007 (Litkowski and Hargraves, 2007).

Initial results in SemEval achieved a best accuracy of 69.3 percent (Ye and Baldwin, 2007). The data from SemEval has subsequently been used in several further investigations of preposition disambiguation. Most notably, Tratz (2011) achieved a result of 88.4 percent accuracy and Srikumar and Roth (2013) achieved a similar result. However, Litkowski (2013b) showed that these results did not extend to other corpora, concluding that the FrameNet-based corpus may not have been representative, with a reduction of accuracy to 39.4 percent using a corpus developed by Oxford.

Litkowski (2013a) announced the creation of the TPP corpora in order to develop a more representative account of preposition behavior. The TPP corpora includes three subcorpora: (1) the full SemEval 2007 corpus (drawn from FrameNet data, henceforth FN), (2) sentences taken from the Oxford English Corpus to exemplify preposition senses in the *Oxford Dictionary of English* (henceforth, OEC), and (3) a sample of sentences drawn from the written portion of the British National Corpus (BNC), using the Word Sketch Engine as implemented in the system for the Corpus Pattern Analysis of verbs (henceforth, CPA or TPP).

We have used the TPP data and the TPP corpora to implement an editorial interface, the Pattern Dictionary of English Prepositions (PDEP).[1] This dictionary is intended to identify the prototypical syntagmatic patterns with which prepositions in use are associated, identifying linguistic units used sequentially to make well-formed structures and characterizing the relationship between these units. In the case of prepositions, the units are the complement (object) of the preposition and the governor (point of attachment) of the prepositional phrase. The editorial interface is used to make changes in the underlying databases, as described in the following subsections. Editorial access to make changes is limited, but the system can be explored publicly and the underlying data can be accessed publicly, either in its entirety or through publicly available scripts used in accessing the data during editorial operations.

Standard dictionaries include definitions of prepositions, but only loosely characterize the syntagmatic patterns associated with each sense. PDEP takes this a step further, looking for prototypical sentence contexts to characterize the patterns. PDEP is modeled on the principles of Corpus Pattern Analysis (CPA), developed to characterize syntagmatic patterns for verbs.[2] These principles are described more fully in Hanks (2013). Currently, CPA is being used in the project Disambiguation of Verbs by Collocation to develop a Pattern Dictionary of English Verbs (PDEV).[3] PDEP is closely related to PDEV, since most syntagmatic patterns for prepositions are related to the main verb in a clause. PDEP is viewed as subordinate to PDEV, sufficiently so that PDEP employs significant portions of code being used in PDEV, with appropriate modifications as necessary to capture the syntagmatic patterns for prepositions.[4]

2.1 The Preposition Inventory

After a start page for entry into PDEP, a table of all prepositions in the sense inventory is displayed. Figure 1 contains a truncated snapshot of this table. The table has a row for each of 304 prepositions as identified in TPP. The second column indicates the number of patterns (senses) for each preposition. The next two columns show the number of TPP (CPA) instances that have been tagged and the total number of TPP instances that have been obtained as the sample from the total number of instances in the BNC.

Preposition ↑	Patterns	TPP Tagged	TPP Insts	BNC Freq
because of	1	250	250	9346
before	4	125	250	20821
behind	9	0	250	6207
below	4	16	250	2646
beneath	6	0	250	1277
beside	3	0	250	932
besides	1	0	250	468
between	9	8	250	58865

Figure 1. Preposition Inventory

Additional columns not shown in Figure 1 show (1) the status of the analysis for the preposition, (2) the number of instances from FrameNet (i.e., FN Insts, as developed for SemEval 2007), and (3) the number of instances from the Oxford English Corpus (i.e., OEC Insts). The number of prepositions with

[1] http://www.clres.com/db/TPPEditor.html

[2] See http://nlp.fi.muni.cz/projects/cpa/.

[3] See http://clg.wlv.ac.uk/projects/DVC

[4] PDEP is implemented as a combination of HTML and Javascript. Within the Javascript code, calls are made to PHP scripts to retrieve data from MySQL database tables and from additional files (described below).

FrameNet instances is 57 (larger than the 34 prepositions used in SemEval). There are no OEC instances for 57 prepositions. There are no TPP instances for 41 prepositions. Notwithstanding the lack of instances, there are TPP characterizations for all 304 prepositions.

The BNC frequency shown in Figure 1 provides a basis for extrapolating results from PDEP to the totality of prepositions. In total, the number of instances in the BNC is 5,391,042, which can be used as the denominator when examining the relative frequency of any preposition (e.g., *between* has a frequency of 0.0109, 58,865/5,391,042).[5]

In general, the target sample size was 250 CPA instances. If the number available was less than 250, all instances were used. The TPP CPA corpus contains 250 instances for 170 prepositions. Where the number of senses for a preposition was large (about 15 or more), larger samples of 750 (*of*, *to*, *on*, and *with*) or 500 (*in*, *for*, *by*, *from*, *at*, *into*, *over*, *like*, and *through*) were drawn.

2.2 Preposition Patterns

When a row in Figure 1 is clicked, the preposition is selected and a new page is opened to show the patterns for that preposition. Figure 2 shows the four patterns for *below*. Each pattern is presented as an instance of the template **[[Governor]] prep [[Complement]]**, followed by its primary implicature, where the current definition is substituted for the preposition.

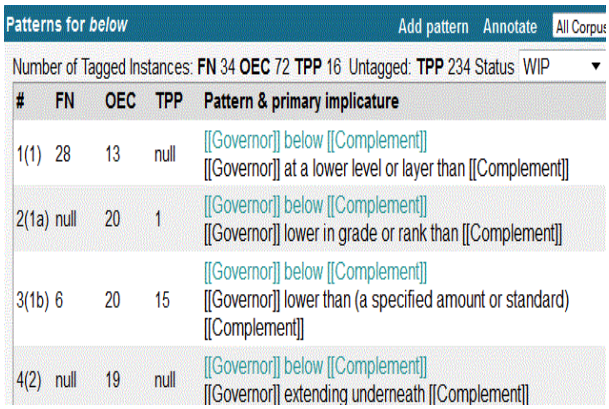


Figure 2. Preposition Pattern List

The display in Figure 2 provides an overview for each preposition, with the top line showing the number of tagged instances available from each corpus. For the TPP instances, this identifies the number of instances that have been tagged and the number that remain to be tagged. In the body of the table, the first column shows the TPP sense number. The next three columns show the number of instances that have been tagged with this sense. Note that the top line of the pattern list includes a menu option for adding a pattern, for the case when we find that a new sense is required by the corpus evidence.

Clicking on any row in the pattern list opens the details for that pattern, with a pattern box entitled with the preposition and the pattern number, as shown in Figure 3. The pattern box contains data developed in TPP and several new fields intended to capture our enhancements.

TPP data include the fields for the **Complement**, the **Governor**, the **TPP Class**, the **TPP Relation**, the **Substitutable Prepositions**, the **Syntactic Position**, the **Quirk Reference**, the **Sense Relation**, and the **Comment**. We have added the checkboxes for complement type (common nouns, proper nouns, WH-phrases, and -ing phrases), as well as a field to identify a particular lexical item (lexset) if the sense is an idiomatic usage. We have added the **Selector** fields for the complement and the governor. For the complement, we have a field **Category** to hold its ontological category (using the shallow ontology being developed for verbs in the DVC project mentioned above).[6] We also provided a field for the **Semantic Class** of the governor; this field has not yet been implemented.

We have added two **Cluster/Relation** fields. The **Cluster** field is based on data available from Tratz (2011), where senses in the SemEval 2007 data have been put into 34 clusters. The **Relation** field is based on data available from Srikumar and Roth (2013), where senses in the SemEval 2007 data have been put into 32 classes. A key element of Srikumar and Roth was the use of these classes to model semantic relations across prepositions (e.g., grouping all the Temporal senses of the SemEval prepositions). In the pattern box, each of these two fields has a drop-down list of the clusters and relations, enabling us to categorize the senses of other prepositions with these classes. Below, we describe how we are able to use the TPP classes and relations along with the Tratz clusters and Srikumar relations in an analysis of these classes across the

[5] The total number of instances for *of* and *in* in this estimate is 1,000,000. As a result, the relative frequency calculation should not be construed as completely accurate.

[6] This ontology is an evolution of the Brandeis Semantic Ontology (Pustejovsky et al., 2006).

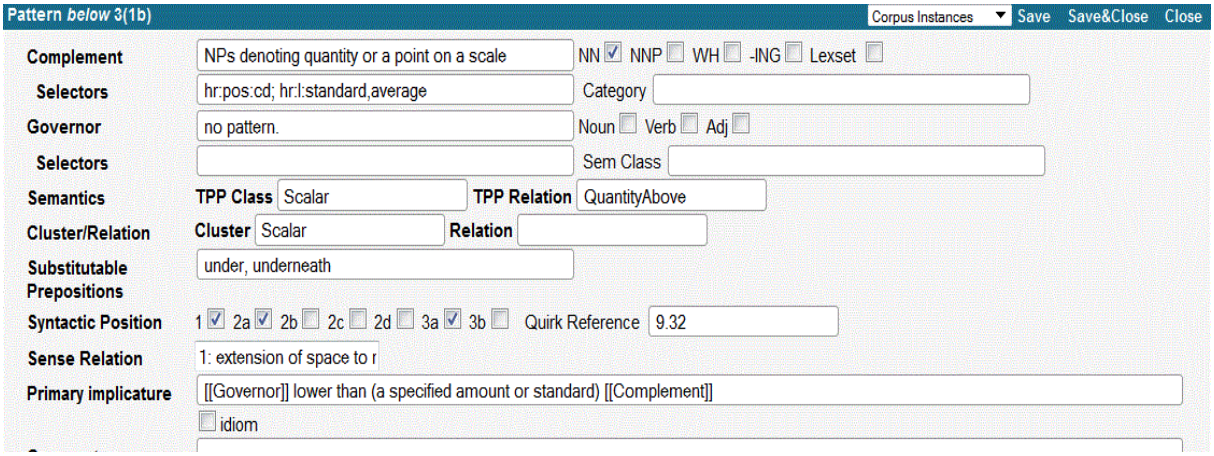


Figure 3. Preposition Pattern Details

full set of prepositions, instead of just those used in SemEval.

Any number of pattern boxes may be opened at one time. The data in any of the fields may be altered (with the menu bar changing color to red) and then saved to the underlying databases. An individual pattern box may then be closed.

The drop-down box labeled **Corpus Instances** in the menu bar is used to open the set of corpus instances for the given sense. As shown in Figure 2, this sense has 6 FN instances, 20 OEC instances, and 15 TPP instances. The drop-down box has an option for each of these sets, along with an option for all TPP instances that have not yet been tagged. When one of these options is selected, the corresponding set of instances is opened in a new tab, discussed in the next section.

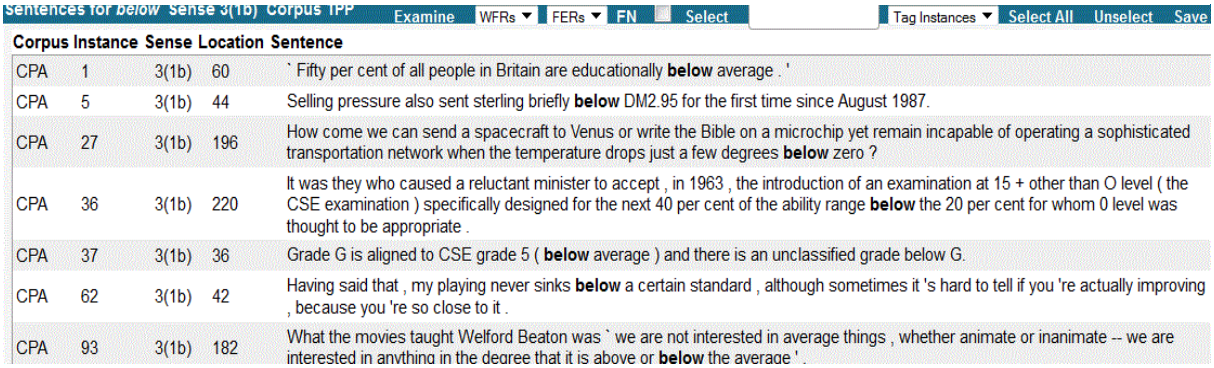
2.3 Preposition Corpus Instances

As indicated, selecting an instance set from the pattern box opens this set in a separate tab, as shown in Figure 4. This tab, labeled **Annotation: below (3(1b))**, identifies the preposition and the sense, if any, associated with the instance set (the sense will be identified as **unk** if the set has not yet been tagged. The instance set is displayed, identifying the corpus, the instance identifier, the TPP sense (if identified, or "unk" if not), the location in the sentence of the target preposition, and the sentence, with the preposition in bold.

This tab is where the annotation takes place. Any set of sentences may be selected; each selected sentence is highlighted in yellow (as shown in Figure 6). The sense value may be changed using the drop-down box labeled **Tag Instances** in the menu bar. This drop-down box contains all the current senses for the preposition, along with possible tags **x** (to indicate that the instance is invalid for the preposition) and **unk** (to indicate that a tagging decision has not yet been made). The sense tags in Figure 4 were originally untagged in the CPA (TPP) corpus and were tagged in this manner.

In general, sense-tagging follows standard lexicographic principles, where an attempt is made to group instances that appear to represent distinct senses. PDEP provides an enhanced environment for this process. Firstly, we can make use of the current TPP sense inventory to tag sentences. Since the pattern sets (definitions) are based on the *Oxford Dictionary of English*, the likelihood that the coverage and accuracy of the sense distinctions is quite high. However, since prepositions have not generally received the close attention of words in other parts of speech,

Figure 4. Preposition Corpus Instance Annotation



Corpus	Instance	Sense	Location	Sentence
CPA	1	3(1b)	60	` Fifty per cent of all people in Britain are educationally **below** average . '
CPA	5	3(1b)	44	Selling pressure also sent sterling briefly **below** DM2.95 for the first time since August 1987.
CPA	27	3(1b)	196	How come we can send a spacecraft to Venus or write the Bible on a microchip yet remain incapable of operating a sophisticated transportation network when the temperature drops just a few degrees **below** zero ?
CPA	36	3(1b)	220	It was they who caused a reluctant minister to accept , in 1963 , the introduction of an examination at 15 + other than O level (the CSE examination) specifically designed for the next 40 per cent of the ability range **below** the 20 per cent for whom 0 level was thought to be appropriate .
CPA	37	3(1b)	36	Grade G is aligned to CSE grade 5 (**below** average) and there is an unclassified grade below G.
CPA	62	3(1b)	42	Having said that , my playing never sinks **below** a certain standard , although sometimes it 's hard to tell if you 're actually improving , because you 're so close to it .
CPA	93	3(1b)	182	What the movies taught Welford Beaton was ` we are not interested in average things , whether animate or inanimate -- we are interested in anything in the degree that it is above or **below** the average ' .

PDEP is intended to ensure the coverage and accuracy. During the tagging of the SemEval instances, the lexicographer found it necessary to increase the number of senses by about 10 percent. Since the lack of coverage of FrameNet is well-recognized, the representative sample developed for the TPP corpus should provide the basis for ensuring the coverage and accuracy.

In addition to adhering to standard lexicographic principles, the availability of the tagged FN and OEC instances can be used as the basis for tagging decisions. Where available, these tagged instances can be opened in separate tabs and used as examples for tagging the unknown TPP instances.

3 Tagging the TPP Corpus

3.1 Examining Corpus Instances

The main contribution of the present work is the ability to interactively examine characteristics of the context surrounding the target preposition in the corpus instances. In the menu bar shown in Figure 4, there is an **Examine** item. Next to it are two drop-down boxes, one labeled **WFRs** (word-finding rules) and one labeled **FERs** (feature extraction rules). These rules are taken from the system described in Tratz and Hovy (2011) and Tratz (2011).[7] The TPP corpora described in Litkowski (2013a) includes full dependency parses and feature files for all sentences. Each sentence may have as many as 1500 features describing the context of the target preposition. We have made the feature files for these sentences (1309 MB) available for exploration in PDEP.

In our system, we make available seven word-finding rules and nine feature extraction rules. The word-finding rules fall into two groups: words pertaining to the governor and words pertaining to the complement. The five governor word-finding rules are (1) verb or head to the left (**l**), (2) head to the left (**hl**), (3) verb to the left (**vl**), (4) word to the left (**wl**), and (5) governor (**h**). The two complement word-finding rules are (1) syntactic preposition complement (**c**) and (2) heuristic preposition complement (**hr**). The feature extraction rules are (1) word class (**wc**), (2) part of speech (**pos**), (3) lemma (**l**), (4) word (**w**), (5) WordNet lexical name (**ln**), (6) WordNet synonyms (**s**), (7) WordNet hypernyms (**h**), (8) whether the word is capitalized (**c**), and (9) affixes (**af**). Thus, we are able to examine any of 63 WFR FER combinations for whatever corpus set happens to be open.

In addition to these features, we are able to determine the extent to which prepositions associated with FrameNet lexical units and VerbNet classes occur in a given corpus set. In Figure 4, there is a checkbox labeled FN next to the FERs drop-down list to examine FrameNet lexical units. There is a similar checkbox labeled VN to examine members of VerbNet classes. These boxes appear only when either of these resources has identified the given preposition as part of its frame (75 for FrameNet and 31 for VerbNet).

When a particular WFR-FER combination is selected and the **Examine** menu item is clicked, a new tab is opened showing the values for those features for the given corpus set, as shown in Figure 5. The tab shows the WFR and FER that were used, the number of features for which the value was found in the feature data, the values, and the count for each feature. The description column is used when displaying results for the part of speech, the affix type, FrameNet frame elements, and VerbNet classes, since the value column for these hits are not self-explanatory. The example in Figure 5 is showing the lemma, which requires no further explanation.

Feature Examination Results for *below*

Word-Finding Rule: *Preposition Complement (Heuristic)*

Feature Extraction Rule: *Lemma*

Number of hits: *13* out of 15 instances

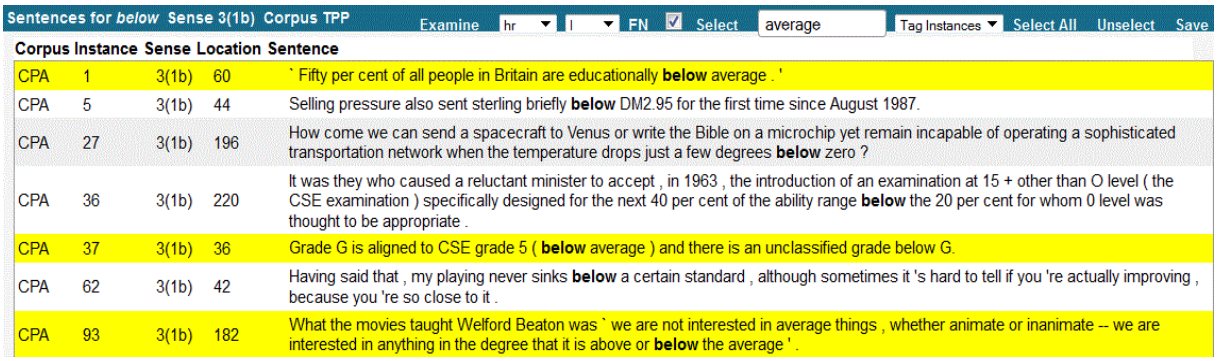
Value	Count	Description
5	1	
20	1	
average	9	
standard	1	
zero	1	

Figure 5. Feature Examination Results

For most features (e.g., lemma or part of speech), the number of possible values is relatively small, limited by the number of instances in the corpus set. For features such as the WordNet lexical name, synonyms and hypernyms, the number of values may be much larger. For FrameNet and VerbNet, the feature examination is limited to the combination of the WFR for the governor (**h**) and the FER lemma (**l**), both of which will generally identify verbs in the value column.

The general objective of examining features is to identify those that are diagnostic of specific senses. When applied to the full untagged TPP corpus set, this process is akin to developing

[7] An updated version of this system is available at http://sourceforge.net/projects/miacp/.



Corpus	Instance	Sense	Location	Sentence
CPA	1	3(1b)	60	` Fifty per cent of all people in Britain are educationally **below** average . '
CPA	5	3(1b)	44	Selling pressure also sent sterling briefly **below** DM2.95 for the first time since August 1987.
CPA	27	3(1b)	196	How come we can send a spacecraft to Venus or write the Bible on a microchip yet remain incapable of operating a sophisticated transportation network when the temperature drops just a few degrees **below** zero ?
CPA	36	3(1b)	220	It was they who caused a reluctant minister to accept , in 1963 , the introduction of an examination at 15 + other than O level (the CSE examination) specifically designed for the next 40 per cent of the ability range **below** the 20 per cent for whom 0 level was thought to be appropriate .
CPA	37	3(1b)	36	Grade G is aligned to CSE grade 5 (**below** average) and there is an unclassified grade below G.
CPA	62	3(1b)	42	Having said that , my playing never sinks **below** a certain standard , although sometimes it 's hard to tell if you 're actually improving , because you 're so close to it .
CPA	93	3(1b)	182	What the movies taught Welford Beaton was ` we are not interested in average things , whether animate or inanimate -- we are interested in anything in the degree that it is above or **below** the average ' .

Figure 6. Selected Corpus Instances

word sketches for prepositions (Kilgarriff et al., 2004). However, since we have tagged corpus sets for most preposition senses, we can begin our efforts looking at these sets. The hypothesis is that the tagged corpora will show patterns which can then be used for tagging instances in the TPP corpus.[8]

The first step in examining features generally is to look at the word classes and parts of speech for the complement and the governor.[9] These are useful for filling in their checkboxes in Figure 3. Another useful feature is word to the left (**wl**), which can be used to verify the syntactic position checkboxes, particularly the adverbial positions (adjunct, subjunct, disjunct, and conjunct). These first steps provide a general overview of a sense's behavior.

The next step of feature examination delves more into the semantic characteristics of the complement and the governor. Tratz (2011) reported that the use of heuristics provided a more accurate identification of the preposition complement; this is the WFR **hr** in our system. After getting some idea of the word class and the part of speech, we next examine the WordNet lexical name of the complement to determine its broad semantic grouping. As mentioned, this feature may return a number of values larger than the size of the corpus set, since WordNet senses for a given lexeme may be polysemous. Notwithstanding, this feature examination generally shows the dominant categories and can be used to characterize and act as a selector for the complement in the pattern details. Similar procedures are used for characterizing the governor selection criteria.

In the example in Figure 3, for *below*, sense *3(1b)*, our preliminary analysis shows **hr:pos:cd** (i.e., a cardinal number) and **hr:l:average, standard** (i.e., the lemmas *average* and *standard*) are particularly useful for identifying this sense.

3.2 Selecting Corpus Instances

In addition to enabling feature examination, PDEP also facilitates selection of corpus instances. We can use the specifications for any WFR - FER combination, along with one of the values (as shown in Figure 5), to select the corpus instances having that feature. Figure 6 shows, in part, the result of the WFR **hr** and FER **l** with the value *average*, against the instances in the open corpus set.

As shown in the menu bar in Figure 6, we can select all instances and unselect all selections. Based on any selections, we can then tag such instances with one of the options that appear in the **Tag Instances** drop-down box. In the specific example, we could change all the selected instances to some other sense, if we have decided that the current assignment is not the best.

The selection mechanism is not used absolutely. For example, in examining the untagged instances for *over*, we used the specification **hr:ln:noun.time** (looking for instances with the heuristic complement having the WordNet lexical name **noun.time**). Out of 500 instances, we found 122 with this property. We then scrolled through the selected items, deselecting instances that did not provide a time period, and then tagged 99 instances with the sense *14(5)*, with the meaning *expressing duration*. Once we have made such a tagging, we can look at just those instances the next time we examine this sense. In this case, we might decide, pace the TPP lexicographer's comment, that the instances should be

[8] Currently, 21.5 percent of the TPP instances (10347 of 47,285) have been tagged.

[9] Accurate identification of the complement and governor is likely improved with the reliance on the Tratz dependency parser. Moreover, this is likely to improve the word sketches in PDEP. Ambati et al. (2012) report that dependency parses provide improved word sketches over purpose-built finite-state grammars. Their findings provide additional support for the methods presented here.

broken down into those which express a time period and those which describe "accompanying circumstances" (e.g., *over coffee*).

3.3 Accuracy of Features

PDEP uses the output from Tratz' system (2011), which is of high quality, but which is not always correct. In addition, the TPP corpus also has some shortcomings, which are revealed in examining the instances. The TPP corpus has not been cleaned in the same manner as the FN and the OEC corpora. As a result, we see many cases which are more difficult to parse and hence, from which to generate feature sets. We believe this provides a truer real-world picture of the complexities of preposition behavior. As a result, in the **Tag Instances** drop-down box, we have included an option to tag a sentence as **x**, to indicate that it is not a valid instance.

A small percentage of the TPP instances are ill-formed, i.e., incomplete sentences; these are marked as **x**. For some prepositions, e.g., *down*, a substantial number of instances are not prepositions, but rather adverbs or particles. For some phrasal prepositions, such as *on the strength of*, the phrase is literal, rather than the preposition idiom; in this case, 20 of 124 instances were marked as **x**. The occurrence of these invalid instances provides an opportunity for improving taggers, parsers, and semantic role labelers.

4 Assessment of Lexical Resources

Since the PDEP system enables exploration of features from WordNet, FrameNet, and VerbNet, we are able to make some assessment of these resources.

WordNet played a statistically significant role in the systems developed by Tratz (2011) and Srikumar and Roth (2013). This includes the WordNet lexicographer's file name (e.g., **noun.time**), synsets, and hypernyms. We make extensive use of the file name, but less so from the synsets and hypernyms. However, in general, we find that the file names are too coarse-grained and the synsets and hypernyms too fine-grained for generalizations on the selectors for the complements and the governors. The issue of granularity also affects the use of the DVC ontology. We discuss this issue further in section 6, on investigations of suitable categorization schemes for PDEP.

In using FrameNet, our results illustrate the unbalanced corpus used in SemEval 2007 (as suggested in Litkowski (2013b)). For the sense of *of*, "used to indicate the contents of a container", we first examined the FrameNet corpus set for that sense, which contains 278 instances (out of 4482, or 6.2 percent). Using PDEP, we found that FrameNet feature values for the governor accounted for 264 of these instances (95 percent), all of which were related to the frame elements *Contents* or *Stuff*. However, in the TPP corpus, only 3 out of 750 instances were identified for this sense (0.4 percent). Thus, while FrameNet culled a large number of instances which had these frame element realizations, these instances do not appear to be representative of their occurrence in a random sample of *of* uses. We have seen similar patterns for the other SemEval prepositions.

A similar situation exists for *Cause* senses of major prepositions: *for* (385 in FrameNet, 5/500 in TPP), *from* (71 in FrameNet, 16/500 in TPP), *of* (68 in FrameNet, 0/750 in TPP), and *with* (127 in FrameNet, 8/750 in TPP). Each of these cases further emphasizes how the SemEval 2007 instances are not representative and thus degrade the ability *to* apply existing preposition disambiguation results beyond these instances.)We discuss *Cause* senses further in the wider context of all PDEP prepositions in the next section on class analyses.)

As indicated earlier, VerbNet identifies fewer prepositions in its frames than FrameNet. We believe this is the case since VerbNet prepositions are generally arguments, rather than adjuncts. Many of the FrameNet prepositions are evoking peripheral and extra-thematic frame elements, so the number of prepositions is correspondingly higher. Also, VerbNet contains fewer members in its verb classes. As a result, the number of hits when using VerbNet is somewhat smaller, although some use of VerbNet classes is possible with the governor selectors.

PDEP provides a vehicle for expanding the items in all these resources. While prepositions are not central to these resources, their supporting role provides additional information that might be useful in developing and using these other resources.

5 Class Analyses

In SemEval 2007, Yuret (2007) investigated the possibility of using the substitutable prepositions as the basis for disambiguation (as part of more general lexical sample substitution). Although his methodology yielded significant gains over the baseline, his best results were only 54.7 per-

cent accuracy, concluding that preposition use is highly idiosyncratic. Srikumar and Roth (2013) broadened this perspective by considering a class-based approach by collapsing semantically-related senses across prepositions, thereby deriving a semantic relation inventory. While their emphasis was on modeling semantic relations, they achieved an accuracy of 83.53 percent for preposition disambiguation.

As mentioned above, PDEP has a field for the Srikumar semantic relation, initially populated for the SemEval prepositions, and being extended to cover all other prepositions. For example, Srikumar and Roth identified 21 temporal senses across 14 SemEval prepositions, while we have thus far identified 62 senses across 50 prepositions. Similar increases in the sizes of other classes occur as well. For causal senses, Srikumar and Roth identified 11 senses over 7 prepositions, while PDEP has 27 senses under 25 prepositions.

PDEP enables an in-depth analysis of TPP classes, Tratz clusters, and Srikumar semantic realations. First, we query the database underlying Figure 3 to identify all senses with a particular class. We then examine each sense on each list in detail.

We follow the procedures laid out above for examining the features to add information about selectors, complement types, and categories. We use this information to tag the TPP instances, conservatively assuring the tagging, e.g., leaving untagged questionable instances. Finally, we carefully place each sense into a preposition class or subclass, grouping senses together and making annotations that attempt to capture any nuance of meaning that distinguishes the sense from other members of the class.

To build a description of the class and its subclasses, we make use of the Quirk reference in Figure 3 (i.e., the relevant discussions in Quirk et al. (1985)). We build the description of a class as a separate web page and make this available as a menu item in Figure 3 (not shown for the Scalar class when that screenshot was made). The description provides an overview of the class, making use of the TPP data and the Quirk discussion, and indicating the number of senses and the number of prepositions. Next, the description provides a list of the categories within the class, characterizing the complements of the category and then listing each sense in the category, with any nuance of meaning as necessary. Finally, we attempt to summarize the selection criteria that have been used across all the senses in the class.

The process of building a class description reveals inconsistencies in each of the class fields. When we place a preposition sense into the class, we may find it necessary to make changes in the underlying data.

At the top level, these class analyses in effect constitute a coarse-grained sense inventory. As the subclasses are developed, a finer-grained analysis of a particular area is available. We believe these analyses may provide a comprehensive characterization of particular semantic roles that can be used for various NLP applications.

6 Availability of PDEP Data and Potential for Further Enhancements

As indicated above, each of the tables shown in the figures is generated in Javascript through a system call to a PHP script. Each of these scripts is described in detail at the PDEP web site. Each script returns data in Javascript Object Notation (JSON), enabling users to obtain whatever data is of interest to them and perhaps using this data dynamically.

While PDEP provides access to a large amount of data, the architecture is very flexible and easy to extend. For this, we are grateful for the Tratz parser and the DVC code.

In building PDEP, we found it necessary to reprocess the SemEval 2007 data of the full 28,052 sentences that were available through TPP, rather than just those that were used in the SemEval task itself. Tagging, parsing, and creating feature files for these sentences took less than 10 minutes, with an equal time to upload the feature files. We would be able to add or substitute new corpora to the PDEP databases with relatively little effort.

Similarly, we can add new elements or modify existing elements that describe preposition patterns. This would require easily-made modifications to the underlying MySQL database tables. The PHP scripts that access these tables are also easily developed or modified. Most of these scripts use less than 100 lines of code.

In developing PDEP, we have added various resources incrementally. This applies to such resources as the DVC ontology, FrameNet, and VerbNet. Each of these resources required relatively little effort to integrate into PDEP. We will continue to investigate the utility of other resources that will assist in characterizing preposition behavior. We have begun to look at the noun clusters used in Srikumar and Roth (2013) for better characterizing complements. We are also

examining an Oxford noun hierarchy as another alternative for complement analysis. We are examining the WordNet detour to FrameNet, as described in Burchardt et al. (2005), particularly for use in further characterizing the governors.

We recognize that an important element of PDEP will be in its utility for preposition disambiguation. While we have not yet begun the necessary experimentation and evaluation, we believe the representativeness and sample sizes of the TPP corpus (mostly with 250 or more sentences per preposition) should provide a basis for constructing the needed studies. We expect that this will follow techniques used by Cinkova et al. (2012), in examining the Pattern Dictionary of English Verbs developed as the precursor to DVC.

We expect that interaction with the NLP community will help PDEP evolve into a useful resource, not only for characterizing preposition behavior, but also for assisting in the development of other lexical resources.

7 Conclusion and Future Plans

We have described the Pattern Dictionary of English Prepositions (PDEP) as a new lexical resource for examining and recording preposition behavior. PDEP does not introduce any ideas that have not already been explored in the investigation of other parts of speech. However, by bringing together work from these disparate sources, we have shown that it is possible to analyze preposition behavior in a manner equivalent to the major parts of speech. Since dictionary publishers have not previously devoted much effort in analyzing preposition behavior, we believe PDEP may serve an important role, particularly for various NLP applications in which semantic role labeling is important.

On the other hand, PDEP as described in this paper is only in its initial stages. In following the principles laid out for verbs in PDEV, a main goal is to provide a sufficient characterization of how frequently different preposition patterns (senses) occur, with some idea of a statistical characterization of the probability of the conjunction of a preposition, its complement, and its governor. Better development of a desired syntagmatic characterization of preposition behavior, consistent with the principles of TNE, is still needed. Since preposition behavior is strongly linked to verb behavior, further effort is needed to link PDEP to PDEV.

The resource will benefit from futher experimentation and evaluation stages. We expect that desired improvements will come from usage in various NLP tasks, particularly word-sense disambiguation and semantic role labeling. In particular, we anticipate that interaction with the NLP community will identify further enhancements, developments, and hints from usage.

Acknowledgments

Stephen Tratz (and Dirk Hovy) provided considerable assistance in using the Tratz parser. Vivek Srikumar graciously provided his data on preposition classes. Vitek Baisa similarly helped with the adaptation of the PDEV Javascript modules. Orin Hargraves, Patrick Hanks, and Eduard Hovy continued to provide valuable insights. Reviewer comments helped sharpen the draft version of the paper.

References

Bharat Ram Ambati, Siva Reddy, and Adam Kilgarriff. 2012. Word Sketches for Turkish. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC)*. Istanbul, 2945-2950.

Aljoscha Burchardt, Katrin Erk, and Anette Frank. 2005. A WordNet Detour to FrameNet. *Proceedings of GLDV workshop GermaNet II*. Bonn.

Silvie Cinkova, Martin Holub, Adam Rambousek, and Lenka Smejkalova. 2012. A database of semantic clusters of verb usages. *Lexical Resources and Evaluation Conference*. Istanbul, 3176-83.

Patrick Hanks. 2004. Corpus Pattern Analysis. In *EURALEX Proceedings.* Vol. I, pp. 87-98. Lorient, France: Université de Bretagne-Sud.

Patrick Hanks. 2013. *Lexical Analysis: Norms and Exploitations*. MIT Press.

Adam Kilgarriff, Pavel Rychly, Pavel Smrz, and David Tugwell. 2004. The Sketch Engine. *Proceedings of EURALEX*. Lorient, France, pp. 105-16.

Ken Litkowski. 2013a. *The Preposition Project Corpora*. Technical Report 13-01. Damascus, MD: CL Research.

Ken Litkowski. 2013b. *Preposition Disambiguation: Still a Problem*. Technical Report 13-02. Damascus, MD: CL Research.

Ken Litkowski and Orin Hargraves. 2005. The preposition project. *ACL-SIGSEM Workshop on "The Linguistic Dimensions of Prepositions and Their Use in Computational Linguistic Formalisms and Applications"*, pages 171–179.

Ken Litkowski and Orin Hargraves. 2006. Coverage and Inheritance in The Preposition Project. In: *Proceedings of the Third ACL-SIGSEM Workshop on Prepositions*. Trento, Italy.ACL. 89-94.

Ken Litkowski and Orin Hargraves. 2007. SemEval-2007 Task 06: Word-Sense Disambiguation of Prepositions. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*, Prague, Czech Republic.

James Pustejovsky, Catherine Havasi, Jessica Littman, Anna Rumshisky, and Marc Verhagen. 2006. Towards a Generative Lexical Resource: The Brandeis Semantic Ontology. *5th Edition of the International Conference on Lexical Resources and Evaluation.*, 1702-5.

Randolph Quirk, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik. 1985. *A Comprehensive Grammar of the English Language*. New York: Longman Inc.

Vivek Srikumar and Dan Roth. 2011. A Joint Model for Extended Semantic Role Labeling. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. ACL, 129-139.

Vivek Srikumar and Dan Roth. 2013. Modeling Semantic Relations Expressed by Prepositions. *Transactions of the Association for Computational Linguistics*, 1.

Angus Stevenson and Catherine Soanes (Eds.). 2003. *The Oxford Dictionary of English*. Oxford: Clarendon Press.

Stephen Tratz. 2011. *Semantically-Enriched Parsing for Natural Language Understanding*. PhD Thesis, University of Southern California.

Stephen Tratz and Eduard Hovy. 2011. A Fast, Accurate, Non-Projective, Semantically-Enriched Parser. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, UK.

Deniz Yuret. 2007. KU: Word Sense Disambiguation by Substitution. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*, Prague, Czech Republic.

Zapirain, B., E. Agirre, L. Marquez, and M. Surdeanu. 2013. Selectional Preferences for Semantic Role Classification. *Computational Linguistics*, 39:3.

Looking at Unbalanced Specialized Comparable Corpora for Bilingual Lexicon Extraction

Emmanuel Morin and **Amir Hazem**
Université de Nantes, LINA UMR CNRS 6241
2 rue de la houssinière, BP 92208, 44322 Nantes Cedex 03, France
{emmanuel.morin,amir.hazem}@univ-nantes.fr

Abstract

The main work in bilingual lexicon extraction from comparable corpora is based on the implicit hypothesis that corpora are balanced. However, the historical context-based projection method dedicated to this task is relatively insensitive to the sizes of each part of the comparable corpus. Within this context, we have carried out a study on the influence of unbalanced specialized comparable corpora on the quality of bilingual terminology extraction through different experiments. Moreover, we have introduced a regression model that boosts the observations of word co-occurrences used in the context-based projection method. Our results show that the use of unbalanced specialized comparable corpora induces a significant gain in the quality of extracted lexicons.

1 Introduction

The bilingual lexicon extraction task from bilingual corpora was initially addressed by using parallel corpora (i.e. a corpus that contains source texts and their translation). However, despite good results in the compilation of bilingual lexicons, parallel corpora are scarce resources, especially for technical domains and for language pairs not involving English. For these reasons, research in bilingual lexicon extraction has focused on another kind of bilingual corpora comprised of texts sharing common features such as domain, genre, sampling period, etc. without having a source text/target text relationship (McEnery and Xiao, 2007). These corpora, well known now as *comparable corpora*, have also initially been introduced as *non-parallel corpora* (Fung, 1995; Rapp, 1995), and *non-aligned corpora* (Tanaka and Iwasaki, 1996). According to Fung and Cheung (2004), who range bilingual corpora from parallel corpora to quasi-comparable corpora going through comparable corpora, there is a continuum from parallel to comparable corpora (i.e. a kind of filiation).

The bilingual lexicon extraction task from comparable corpora inherits this filiation. For instance, the historical context-based projection method (Fung, 1995; Rapp, 1995), known as the *standard approach*, dedicated to this task seems implicitly to lead to work with balanced comparable corpora in the same way as for parallel corpora (i.e. each part of the corpus is composed of the same amount of data).

In this paper we want to show that the assumption that comparable corpora should be balanced for bilingual lexicon extraction task is unfounded. Moreover, this assumption is prejudicial for specialized comparable corpora, especially when involving the English language for which many documents are available due the prevailing position of this language as a standard for international scientific publications. Within this context, our main contribution consists in a re-reading of the standard approach putting emphasis on the unfounded assumption of the balance of the specialized comparable corpora. In specialized domains, the comparable corpora are traditionally of small size (around 1 million words) in comparison with comparable corpus-based general language (up to 100 million words). Consequently, the observations of word co-occurrences which is the basis of the standard approach are unreliable. To make them more reliable, our second contribution is to contrast different regression models in order to boost the observations of word co-occurrences. This strategy allows to improve the quality of extracted bilingual lexicons from comparable corpora.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1284–1293,
Baltimore, Maryland, USA, June 23-25 2014.

2 Bilingual Lexicon Extraction

In this section, we first describe the standard approach that deals with the task of bilingual lexicon extraction from comparable corpora. We then present an extension of this approach based on regression models. Finally, we discuss works related to this study.

2.1 Standard Approach

The main work in bilingual lexicon extraction from comparable corpora is based on lexical context analysis and relies on the simple observation that a word and its translation tend to appear in the same lexical contexts. The basis of this observation consists in the identification of "first-order affinities" for each source and target language: "*First-order affinities describe what other words are likely to be found in the immediate vicinity of a given word*" (Grefenstette, 1994, p. 279). These affinities can be represented by context vectors, and each vector element represents a word which occurs within the window of the word to be translated (e.g. a seven-word window approximates syntactic dependencies). In order to emphasize significant words in the context vector and to reduce word-frequency effects, the context vectors are normalized according to an association measure. Then, the translation is obtained by comparing the source context vector to each translation candidate vector after having translated each element of the source vector with a general dictionary.

The implementation of the standard approach can be carried out by applying the following three steps (Rapp, 1999; Chiao and Zweigenbaum, 2002; Déjean et al., 2002; Morin et al., 2007; Laroche and Langlais, 2010, among others):

Computing context vectors We collect all the words in the context of each word i and count their occurrence frequency in a window of n words around i. For each word i of the source and the target languages, we obtain a context vector v_i which gathers the set of co-occurrence words j associated with the number of times that j and i occur together $cooc(i, j)$. In order to identify specific words in the lexical context and to reduce word-frequency effects, we normalize context vectors using an association score such as Mutual Information, Log-likelihood, or the discounted log-odds (LO) (Evert, 2005) (see equation 1 and Table 1 where $N = a + b + c + d$).

Transferring context vectors Using a bilingual dictionary, we translate the elements of the source context vector. If the bilingual dictionary provides several translations for an element, we consider all of them but weight the different translations according to their frequency in the target language.

Finding candidate translations For a word to be translated, we compute the similarity between the translated context vector and all target vectors through vector distance measures such as Jaccard or Cosine (see equation 2 where $assoc^i_j$ stands for "association score", v_k is the transferred context vector of the word k to translate, and v_l is the context vector of the word l in the target language). Finally, the candidate translations of a word are the target words ranked following the similarity score.

	j	$\neg j$
i	$a = cooc(i, j)$	$b = cooc(i, \neg j)$
$\neg i$	$c = cooc(\neg i, j)$	$d = cooc(\neg i, \neg j)$

Table 1: Contingency table

$$LO(i, j) = \log \frac{(a + \frac{1}{2}) \times (d + \frac{1}{2})}{(b + \frac{1}{2}) \times (c + \frac{1}{2})} \quad (1)$$

$$Cosine^{v_k}_{v_l} = \frac{\sum_t assoc^l_t \, assoc^k_t}{\sqrt{\sum_t {assoc^l_t}^2} \sqrt{\sum_t {assoc^k_t}^2}} \quad (2)$$

This approach is sensitive to the choice of parameters such as the size of the context, the choice of the association and similarity measures. The most complete study about the influence of these parameters on the quality of word alignment has been carried out by Laroche and Langlais (2010).

The standard approach is used by most researchers so far (Rapp, 1995; Fung, 1998; Peters and Picchi, 1998; Rapp, 1999; Chiao and Zweigenbaum, 2002; Déjean et al., 2002; Gaussier et al., 2004; Morin et al., 2007; Laroche and Langlais, 2010; Prochasson and Fung, 2011;

References	Domain	Languages	Source/Target Sizes
Tanaka and Iwasaki (1996)	Newspaper	EN/JP	30/33 million words
Fung and McKeown (1997)	Newspaper	EN/JP	49/60 million bytes of data
Rapp (1999)	Newspaper	GE/EN	135/163 million words
Chiao and Zweigenbaum (2002)	Medical	FR/EN	602,484/608,320 words
Déjean et al. (2002)	Medical	GE/EN	100,000/100,000 words
Morin et al. (2007)	Medical	FR/JP	693,666/807,287 words
Otero (2007)	European Parliament	SP/EN	14/17 million words
Ismail and Manandhar (2010)	European Parliament	EN/SP	500,000/500,000 sentences
Bouamor et al. (2013)	Financial	FR/EN	402,486/756,840 words
-	Medical	FR/EN	396,524/524,805 words

Table 2: Characteristics of the comparable corpora used for bilingual lexicon extraction

Bouamor et al., 2013, among others) with the implicit hypothesis that comparable corpora are balanced. As McEnery and Xiao (2007, p. 21) observe, a specialized comparable corpus is built as balanced by analogy with a parallel corpus: "*Therefore, in relation to parallel corpora, it is more likely for comparable corpora to be designed as general balanced corpora.*". For instance, Table 2 describes the comparable corpora used in the main work dedicated to bilingual lexicon extraction for which the ratio between the size of the source and the target texts is comprised between 1 and 1.8.

In fact, the assumption that words which have the same meaning in different languages should have the same lexical context distributions does not involve working with balanced comparable corpora. To our knowledge, no attention[1] has been paid to the problem of using unbalanced comparable corpora for bilingual lexicon extraction. Since the context vectors are computed from each part of the comparable corpus rather than through the parts of the comparable corpora, the standard approach is relatively insensitive to differences in corpus sizes. The only precaution for using the standard approach with unbalanced corpora is to normalize the association measure (for instance, this can be done by dividing each entry of a given context vector by the sum of its association scores).

2.2 Prediction Model

Since comparable corpora are usually small in specialized domains (see Table 2), the discriminative power of context vectors (i.e. the observations of word co-occurrences) is reduced. One way to deal with this problem is to re-estimate co-occurrence counts by a prediction function (Hazem and Morin, 2013). This consists in assigning to each observed co-occurrence count of a small comparable corpora, a new value learned beforehand from a large training corpus.

In order to make co-occurrence counts more discriminant and in the same way as Hazem and Morin (2013), one strategy consists in addressing this problem through regression: given training corpora of small and large size (abundant in the general domain), we predict word co-occurrence counts in order to make them more reliable. We then apply the resulting regression function to each word co-occurrence count as a pre-processing step of the standard approach. Our work differs from Hazem and Morin (2013) in two ways. First, while they experienced the linear regression model, we propose to contrast different regression models. Second, we apply regression to unbalanced comparable corpora and study the impact of prediction when applied to the source texts, the target texts and both source and target texts of the used comparable corpora.

We use regression analysis to describe the relationship between word co-occurrence counts in a large corpus (the response variable) and word co-occurrence counts in a small corpus (the predictor variable). As most regression models have already been described in great detail (Christensen, 1997; Agresti, 2007), the derivation of most models is only briefly introduced in this work.

As we can not claim that the prediction of word co-occurrence counts is a linear problem, we consider in addition to the simple linear regression

[1]We only found mention of this aspect in Diab and Finch (2000, p. 1501) "*In principle, we do not have to have the same size corpora in order for the approach to work*".

model (Lin), a generalized linear model which is the logistic regression model ($Logit$) and non linear regression models such as polynomial regression model ($Poly^n$) of order n. Given an input vector $x \in \mathbb{R}^m$, where x_1,...,x_m represent features, we find a prediction $\hat{y} \in \mathbb{R}^m$ for the co-occurrence count of a couple of words $y \in \mathbb{R}$ using one of the regression models presented below:

$$\hat{y}_{Lin} = \beta_0 + \beta_1 x \quad (3)$$

$$\hat{y}_{Logit} = \frac{1}{1 + \exp(-(\beta_0 + \beta_1 x))} \quad (4)$$

$$\hat{y}_{Poly^n} = \beta_0 + \beta_1 x + \beta_2 x^2 + ... + \beta_n x^n \quad (5)$$

where β_i are the parameters to estimate.

Let us denote by f the regression function and by $cooc(w_i, w_j)$ the co-occurrence count of the words w_i and w_j. The resulting predicted value of $cooc(w_i, w_j)$, noted $\hat{cooc}(w_i, w_j)$ is given by the following equation:

$$\hat{cooc}(w_i, w_j) = f(cooc(w_i, w_j)) \quad (6)$$

2.3 Related Work

In the past few years, several contributions have been proposed to improve each step of the standard approach.

Prochasson et al. (2009) enhance the representativeness of the context vector by strengthening the context words that happen to be transliterated words and scientific compound words in the target language. Ismail and Manandhar (2010) also suggest that context vectors should be based on the most important contextually relevant words (in-domain terms), and thus propose a method for filtering the noise of the context vectors. In another way, Rubino and Linarès (2011) improve the context words based on the hypothesis that a word and its candidate translations share thematic similarities. Yu and Tsujii (2009) and Otero (2007) propose, for their part, to replace the window-based method by a syntax-based method in order to improve the representation of the lexical context.

To improve the transfer context vectors step, and increase the number of elements of translated context vectors, Chiao and Zweigenbaum (2003) and Morin and Prochasson (2011) combine a standard general language dictionary with a specialized dictionary, whereas Déjean et al. (2002) use the hierarchical properties of a specialized thesaurus. Koehn and Knight (2002) automatically induce the initial seed bilingual dictionary by using identical spelling features such as cognates and similar contexts. As regards the problem of words ambiguities, Bouamor et al. (2013) carried out word sense disambiguation process only in the target language whereas Gaussier et al. (2004) solve the problem through the source and target languages by using approaches based on CCA (Canonical Correlation Analysis) and multilingual PLSA (Probabilistic Latent Semantic Analysis).

The rank of candidate translations can be improved by integrating different heuristics. For instance, Chiao and Zweigenbaum (2002) introduce a heuristic based on word distribution symmetry. From the ranked list of candidate translations, the standard approach is applied in the reverse direction to find the source counterparts of the first target candidate translations. And then only the target candidate translations that had the initial source word among the first reverse candidate translations are kept. Laroche and Langlais (2010) suggest a heuristic based on the graphic similarity between source and target terms. Here, candidate translations which are cognates of the word to be translated are ranked first among the list of translation candidates.

3 Linguistic Resources

In this section, we outline the different textual resources used for our experiments: the comparable corpora, the bilingual dictionary and the terminology reference lists.

3.1 Specialized Comparable Corpora

For our experiments, we used two specialized French/English comparable corpora:

Breast cancer corpus This comparable corpus is composed of documents collected from the Elsevier website[2]. The documents were taken from the medical domain within the subdomain of "breast cancer". We have automatically selected the documents published between 2001 and 2008 where the title or the keywords contain the term *cancer du sein* in French and *breast cancer* in English. We collected 130 French documents (about 530,000 words) and 1,640 English documents (about

[2] http://www.elsevier.com

7.4 million words). We split the English documents into 14 parts each containing about 530,000 words.

Diabetes corpus The documents making up the French part of the comparable corpus have been craweled from the web using three keywords: *diabète* (diabetes), *alimentation* (food), and *obésité* (obesity). After a manual selection, we only kept the documents which were relative to the medical domain. As a result, 65 French documents were extracted (about 257,000 words). The English part has been extracted from the medical website PubMed[3] using the keywords: *diabetes*, *nutrition* and *feeding*. We only kept the free fulltext available documents. As a result, 2,339 English documents were extracted (about 3,5 million words). We also split the English documents into 14 parts each containing about 250,000 words.

The French and English documents were then normalised through the following linguistic preprocessing steps: tokenisation, part-of-speech tagging, and lemmatisation. These steps were carried out using the TTC TermSuite[4] that applies the same method to several languages including French and English. Finally, the function words were removed and the words occurring less than twice in the French part and in each English part were discarded. Table 3 shows the number of distinct words (# words) after these steps. It also indicates the comparability degree in percentage (comp.) between the French part and each English part of each comparable corpus. The comparability measure (Li and Gaussier, 2010) is based on the expectation of finding the translation for each word in the corpus and gives a good idea about how two corpora are comparable. We can notice that all the comparable corpora have a high degree of comparability with a better comparability of the breast cancer corpora as opposed to the diabetes corpora. In the remainder of this article, [breast cancer corpus i] for instance stands for the breast cancer comparable corpus composed of the unique French part and the English part i ($i \in [1, 14]$).

3.2 Bilingual Dictionary

The bilingual dictionary used in our experiments is the French/English dictionary ELRA-M0033 available from the ELRA catalogue[5]. This resource is a general language dictionary which contains only a few terms related to the medical domain.

	Breast cancer # words (comp.)	Diabetes # words (comp.)
French		
Part 1	7,376	4,982
English		
Part 1	8,214 (79.2)	5,181 (75.2)
Part 2	7,788 (78.8)	5,446 (75.9)
Part 3	8,370 (78.8)	5,610 (76.6)
Part 4	7,992 (79.3)	5,426 (74.8)
Part 5	7,958 (78.7)	5,610 (75.0)
Part 6	8,230 (79.1)	5,719 (73.6)
Part 7	8,035 (78.3)	5,362 (75.6)
Part 8	8,008 (78.8)	5,432 (74.6)
Part 9	8,334 (79.6)	5,398 (74.2)
Part 10	7,978 (79.1)	5,059 (75.6)
Part 11	8,373 (79.4)	5,264 (74.9)
Part 12	8,065 (78.9)	4,644 (73.4)
Part 13	7,847 (80.0)	5,369 (74.8)
Part 14	8,457 (78.9)	5,669 (74.8)

Table 3: Number of distinct words (# words) and degree of comparability (comp.) for each comparable corpora

3.3 Terminology Reference Lists

To evaluate the quality of terminology extraction, we built a bilingual terminology reference list for each comparable corpus. We selected all French/English single words from the UMLS[6] meta-thesaurus. We kept only i) the French single words which occur more than four times in the French part and ii) the English single words which occur more than four times in each English part i[7]. As a result of filtering, 169 French/English single words were extracted for the breast cancer corpus and 244 French/English single words were extracted for the diabetes corpus. It should be noted that the evaluation of terminology extraction using specialized comparable corpora of-

[3] http://www.ncbi.nlm.nih.gov/pubmed/

[4] http://code.google.com/p/ttc-project

[5] http://www.elra.info/

[6] http://www.nlm.nih.gov/research/umls

[7] The threshold sets to four is required to build a bilingual terminology reference list composed of about a hundred words. This value is very low to obtain representative context vectors. For instance, Prochasson and Fung (2011) showed that the standard approach is not relevant for infrequent words (since the context vectors are very unrepresentative i.e. poor in information).

	Breast cancer corpus													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Balanced	26.1	26.2	21.0	27.0	22.8	27.1	26.3	25.8	29.2	23.3	21.7	**29.6**	29.1	26.1
Unbalanced	26.1	31.9	34.7	36.0	37.7	36.4	36.6	37.2	39.8	40.5	40.6	**42.3**	40.9	41.6
	Diabetes corpus													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Balanced	13.6	13.5	11.9	14.6	14.6	11.0	**16.5**	10.5	12.9	13.3	15.2	11.8	13.0	14.3
Unbalanced	13.6	17.5	18.9	21.2	23.4	23.8	24.8	24.7	24.7	24.4	24.8	25.2	**26.0**	24.9

Table 4: Results (MAP %) of the standard approach using the balanced and unbalanced comparable corpora

ten relies on lists of a small size: 95 single words in Chiao and Zweigenbaum (2002), 100 in Morin et al. (2007), 125 and 79 in Bouamor et al. (2013).

4 Experiments and Results

In this section, we present experiments to evaluate the influence of comparable corpus size and prediction models on the quality of bilingual terminology extraction.

We present the results obtained for the terms belonging to the reference list for English to French direction measured in terms of the Mean Average Precision (MAP) (Manning et al., 2008) as follows:

$$MAP(Ref) = \frac{1}{|Ref|} \sum_{i=1}^{|Ref|} \frac{1}{r_i} \quad (7)$$

where $|Ref|$ is the number of terms of the reference list and r_i the rank of the correct candidate translation i.

4.1 Standard Approach Evaluation

In order to evaluate the influence of corpus size on the bilingual terminology extraction task, two experiments have been carried out using the standard approach. We first performed an experiment using each comparable corpus independently of the others (we refer to these corpora as balanced corpora). We then conducted a second experiment where we varied the size of the English part of the comparable corpus, from 530,000 to 7.4 million words for the breast cancer corpus in 530,000 words steps, and from 250,000 to 3.5 million words for the diabetes corpus in 250,000 words steps (we refer to these corpora as unbalanced corpora). In the experiments reported here, the size of the context window w was set to 3 (i.e. a seven-word window that approximates syntactic dependencies), the retained association and similarity measures were the discounted log-odds and the Cosine (see Section 2.1). The results shown were those that give the best performance for the comparable corpora used individually.

Table 4 shows the results of the standard approach on the balanced and the unbalanced breast cancer and diabetes comparable corpora. Each column corresponds to the English part i ($i \in [1, 14]$) of a given comparable corpus. The first line presents the results for each individual comparable corpus and the second line presents the results for the cumulative comparable corpus. For instance, the column 3 indicates the MAP obtained by using a comparable corpus that is composed i) only of [breast cancer corpus 3] (MAP of 21.0%), and ii) of [breast cancer corpus 1, 2 and 3] (MAP of 34.7%).

As a preliminary remark, we can notice that the results differ noticeably according to the comparable corpus used individually (MAP variation between 21.0% and 29.6% for the breast cancer corpora and between 10.5% and 16.5% for the diabetes corpora). We can also note that the MAP of all the unbalanced comparable corpora is always higher than any individual comparable corpus. Overall, starting with a MAP of 26.1% as provided by the balanced [breast cancer corpus 1], we are able to increase it to 42.3% with the unbalanced [breast cancer corpus 12] (the variation observed for some unbalanced corpora such as [diabetes corpus 12, 13 and 14] can be explained by the fact that adding more data in the source language increases the error rate of the translation phase of the standard approach, which leads to the introduction of additional noise in the translated context vectors).

	Balanced breast cancer corpus													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$No\ prediction$	26.1	26.2	21.0	27.0	22.8	27.1	26.3	25.8	29.2	23.3	21.7	**29.6**	29.1	26.1
$Source_{pred}$	26.5	26.0	23.0	30.0	25.4	30.1	28.3	29.4	**32.1**	24.9	24.4	30.5	30.1	29.0
$Target_{pred}$	19.5	20.0	17.2	23.4	19.9	23.1	21.4	21.6	24.1	19.3	18.1	**26.6**	24.3	22.6
$Source_{pred} + Target_{pred}$	23.9	21.9	20.5	25.8	23.5	25.3	24.1	26.1	27.4	22.5	21.0	25.6	**28.5**	24.6
	Balanced diabetes corpus													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$No\ prediction$	13.6	13.5	11.9	14.6	14.6	11.0	**16.5**	10.5	12.9	13.3	15.2	11.8	13.0	14.3
$Source_{pred}$	13.9	14.3	12.6	15.5	14.9	10.9	**17.6**	11.1	14.0	14.2	16.4	13.3	13.5	15.7
$Target_{pred}$	09.8	09.0	08.3	11.9	10.1	08.0	**15.9**	07.3	10.8	10.0	10.1	08.8	10.8	10.2
$Source_{pred} + Target_{pred}$	10.9	11.0	09.0	13.6	11.8	08.6	**15.4**	07.7	12.8	11.5	11.9	10.5	11.7	11.8

Table 5: Results (MAP %) of the standard approach using the Lin regression model on the balanced breast cancer and diabetes corpora (comparison of predicting the source side, the target side and both sides of the comparable corpora)

4.2 Prediction Evaluation

The aim of this experiment is two-fold: first, we want to evaluate the usefulness of predicting word co-occurrence counts and second, we want to find out whether it is more appropriate to apply prediction to the source side, the target side or both sides of the bilingual comparable corpora.

	Breast cancer	Diabetes
$No\ prediction$	29.6	16.5
Lin	30.5	**17.6**
$Poly^2$	**30.6**	17.5
$Poly^3$	30.4	**17.6**
$Logit$	22.3	13.6

Table 6: Results (MAP %) of the standard approach using different regression models on the balanced breast cancer and diabetes corpora

4.2.1 Regression Models Comparison

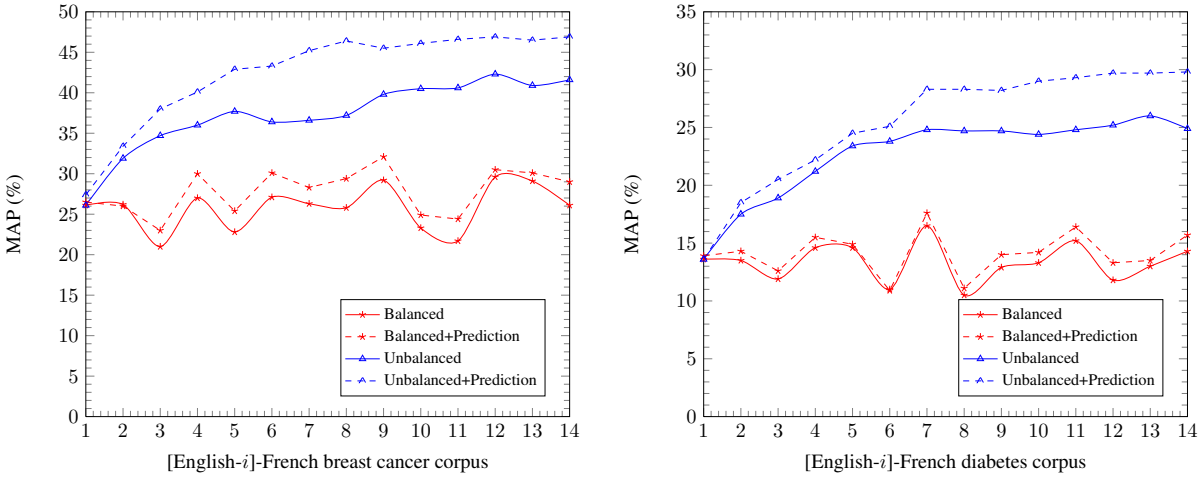
We contrast the prediction models presented in Section 2.2 to findout which is the most appropriate model to use as a pre-processing step of the standard approach. We chose the balanced corpora where the standard approach has shown the best results in the previous experiment, namely [breast cancer corpus 12] and [diabetes corpus 7].

Table 6 shows a comparison between the standard approach without prediction noted $No\ prediction$ and the standard approach with prediction models. We contrast the simple linear regression model (Lin) with the second and the third order polynomial regressions ($Poly^2$ and $Poly^3$) and the logistic regression model ($Logit$). We can notice that except for the $Logit$ model, all the regression models outperform the baseline ($No\ prediction$). Also, as we can see, the results obtained with the linear and polynomial regressions are very close. This suggests that both linear and polynomial regressions are suitable as a pre-processing step of the standard approach, while the logistic regression seems to be inappropriate according to the results shown in Table 6.

That said, the gain of regression models is not significant. This may be due to the regression parameters that have been learned from a training corpus of the general domain. Another reason that could explain these results is the prediction process. We applied the same regression function to all co-occurrence counts while learning models for low and high frequencies should have been more appropriate. In the light of the above results, we believe that prediction can be beneficial to our task.

4.2.2 Source versus Target Prediction

Table 5 shows a comparison between the standard approach without prediction noted $No\ prediction$ and the standard approach based on the prediction of the source side noted $Source_{pred}$, the target side noted $Target_{pred}$ and both sides noted $Source_{pred}$+$Target_{pred}$. If prediction can not replace a large amount of data, it aims at increasing co-occurrence counts as if large amounts of data were at our disposal. In this case, applying prediction to the source side may simulate a configuration of using unbalanced comparable corpora where the source side is n times bigger than the target side. Predicting the target side only, may



(a) (b)

Figure 1: Results (MAP %) of the standard approach using the best configurations of the prediction models (Lin for $Balanced + Prediction$ and $Poly^2$ for $Unbalanced + Prediction$) on the breast cancer and the diabetes corpora

leads us to the opposite configuration where the target side is n times bigger than the source side. Finally, predicting both sides may simulate a large comparable corpora on both sides. In this experiment, we chose to use the linear regression model (Lin) for the prediction part. That said, the other regression models have shown the same behavior as Lin.

We can see that the best results are obtained by the $Source_{pred}$ approach for both comparable corpora. We can also notice that predicting the target side and both sides of the comparable corpora degrades the results. It is not surprising that predicting the target side only leads to lower results, since it is well known that a better characterization of a word to translate (given from the source side) leads to better results. We can deduce from Table 5 that source prediction is the most appropriate configuration to improve the quality of extracted lexicons. This configuration which simulates the use of unbalanced corpora leads us to think that using prediction with unbalanced comparable corpora should also increase the performance of the standard approach. This assumption is evaluated in the next Subsection.

4.3 Predicting Unbalanced Corpora

In this last experiment we contrast the standard approach applied to the balanced and unbalanced corpora noted $Balanced$ and $Unbalanced$ with the standard approach combined with the prediction model noted $Balanced + Prediction$ and $Unbalanced + Prediction$.

Figure 1(a) illustrates the results of the experiments conducted on the breast cancer corpus. We can see that the $Unbalanced$ approach significantly outperforms the baseline ($Balanced$). The big difference between the $Balanced$ and the $Unbalanced$ approaches would indicate that the latter is optimal. We can also notice that the prediction model applied to the balanced corpus ($Balanced + Prediction$) slightly outperforms the baseline while the $Unbalanced + Prediction$ approach significantly outperforms the three other approaches (moreover the variation observed with the $Unbalanced$ approach are lower than the $Unbalanced + Prediction$ approach). Overall, the prediction increases the performance of the standard approach especially for unbalanced corpora.

The results of the experiments conducted on the diabetes corpus are shown in Figure 1(b). As for the previous experiment, we can see that the $Unbalanced$ approach significantly outperforms the $Balanced$ approach. This confirms the unbalanced hypothesis and would motivate the use of unbalanced corpora when they are available. We can also notice that the $Balanced + Prediction$ approach slightly outperforms the baseline while the $Unbalanced + Prediction$ approach gives the best results. Here also, the prediction increases the performance of the standard approach especially for unbalanced corpora. It is clear that in addition to the benefit of using unbalanced comparable

corpora, prediction shows a positive impact on the performance of the standard approach.

5 Conclusion

In this paper, we have studied how an unbalanced specialized comparable corpus could influence the quality of the bilingual lexicon extraction. This aspect represents a significant interest when working with specialized comparable corpora for which the quantity of the data collected may differ depending on the languages involved, especially when involving the English language as many scientific documents are available. More precisely, our different experiments show that using an unbalanced specialized comparable corpus always improves the quality of word translations. Thus, the MAP goes up from 29.6% (best result on the balanced corpora) to 42.3% (best result on the unbalanced corpora) in the breast cancer domain, and from 16.5% to 26.0% in the diabetes domain. Additionally, these results can be improved by using a prediction model of the word co-occurrence counts. Here, the MAP goes up from 42.3% (best result on the unbalanced corpora) to 46.9% (best result on the unbalanced corpora with prediction) in the breast cancer domain, and from 26.0% to 29.8% in the diabetes domain. We hope that this study will pave the way for using specialized unbalanced comparable corpora for bilingual lexicon extraction.

Acknowledgments

This work is supported by the French National Research Agency under grant ANR-12-CORD-0020.

References

Alan Agresti. 2007. *An Introduction to Categorical Data Analysis (2nd ed.)*. Wiley & Sons, Inc., Hoboken, New Jersey.

Dhouha Bouamor, Nasredine Semmar, and Pierre Zweigenbaum. 2013. Context vector disambiguation for bilingual lexicon extraction from comparable corpora. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL'13)*, pages 759–764, Sofia, Bulgaria.

Yun-Chuang Chiao and Pierre Zweigenbaum. 2002. Looking for candidate translational equivalents in specialized, comparable corpora. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*, pages 1208–1212, Tapei, Taiwan.

Yun-Chuang Chiao and Pierre Zweigenbaum. 2003. The Effect of a General Lexicon in Corpus-Based Identification of French-English Medical Word Translations. In *The New Navigators: from Professionals to Patients, Actes Medical Informatics Europe*, pages 397–402.

Ronald Christensen. 1997. *Log-Linear Models and Logistic Regression*. Springer-Verlag, Berlin.

Hervé Déjean, Fatia Sadat, and Éric Gaussier. 2002. An approach based on multilingual thesauri and model combination for bilingual lexicon extraction. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*, pages 218–224, Tapei, Taiwan.

Mona T. Diab and Steve Finch. 2000. A Statistical Word-Level Translation Model for Comparable Corpora. In *Proceedings of the 6th International Conference on Computer-Assisted Information Retrieval (RIAO'00)*, pages 1500–1501, Paris, France.

Stefan Evert. 2005. *The Statistics of Word Cooccurrences: Word Pairs and Collocations*. Ph.D. thesis, Universität Stuttgart, Germany.

Pascale Fung and Percy Cheung. 2004. Multi-level bootstrapping for extracting parallel sentences from a quasi-comparable corpus. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING'04)*, pages 1051–1057, Geneva, Switzerland.

Pascale Fung and Kathleen McKeown. 1997. Finding Terminology Translations from Non-parallel Corpora. In *Proceedings of the 5th Annual Workshop on Very Large Corpora (VLC'97)*, pages 192–202, Hong Kong.

Pascale Fung. 1995. Compiling Bilingual Lexicon Entries from a non-Parallel English-Chinese Corpus. In *Proceedings of the 3rd Annual Workshop on Very Large Corpora (VLC'95)*, pages 173–183, Cambridge, MA, USA.

Pascale Fung. 1998. A Statistical View on Bilingual Lexicon Extraction: From Parallel Corpora to Non-parallel Corpora. In David Farwell, Laurie Gerber, and Eduard Hovy, editors, *Proceedings of the 3rd Conference of the Association for Machine Translation in the Americas (AMTA'98)*, pages 1–16, Langhorne, PA, USA.

Éric Gaussier, Jean-Michel Renders, Irena Matveeva, Cyril Goutte, and Hervé Déjean. 2004. A Geometric View on Bilingual Lexicon Extraction from Comparable Corpora. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL'04)*, pages 526–533, Barcelona, Spain.

Gregory Grefenstette. 1994. Corpus-Derived First, Second and Third-Order Word Affinities. In *Proceedings of the 6th Congress of the European Association for Lexicography (EURALEX'94)*, pages 279–290, Amsterdam, The Netherlands.

Amir Hazem and Emmanuel Morin. 2013. Word co-occurrence counts prediction for bilingual terminology extraction from comparable corpora. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing (IJCNLP'13)*, pages 1392–1400, Nagoya, Japan.

Azniah Ismail and Suresh Manandhar. 2010. Bilingual lexicon extraction from comparable corpora using in-domain terms. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING'10)*, pages 481–489, Beijing, China.

Philipp Koehn and Kevin Knight. 2002. Learning a translation lexicon from monolingual corpora. In *Proceedings of the ACL-02 Workshop on Unsupervised Lexical Acquisition (ULA'02)*, pages 9–16, Philadelphia, PA, USA.

Audrey Laroche and Philippe Langlais. 2010. Revisiting Context-based Projection Methods for Term-Translation Spotting in Comparable Corpora. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING'10)*, pages 617–625, Beijing, China.

Bo Li and Éric Gaussier. 2010. Improving corpus comparability for bilingual lexicon extraction from comparable corpora. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING'10)*, pages 644–652, Beijing, China.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Anthony McEnery and Zhonghua Xiao. 2007. Parallel and comparable corpora: What are they up to? In Gunilla Anderman and Margaret Rogers, editors, *Incorporating Corpora: Translation and the Linguist*, Multilingual Matters, chapter 2, pages 18–31. Clevedon, UK.

Emmanuel Morin and Emmanuel Prochasson. 2011. Bilingual lexicon extraction from comparable corpora enhanced with parallel corpora. In *Proceedings of the 4th Workshop on Building and Using Comparable Corpora (BUCC'11)*, pages 27–34, Portland, OR, USA.

Emmanuel Morin, Béatrice Daille, Koichi Takeuchi, and Kyo Kageura. 2007. Bilingual Terminology Mining – Using Brain, not brawn comparable corpora. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL'07)*, pages 664–671, Prague, Czech Republic.

Pablo Gamallo Otero. 2007. Learning bilingual lexicons from comparable english and spanish corpora. In *Proceedings of the 11th Conference on Machine Translation Summit (MT Summit XI)*, pages 191–198, Copenhagen, Denmark.

Carol Peters and Eugenio Picchi. 1998. Cross-language information retrieval: A system for comparable corpus querying. In Gregory Grefenstette, editor, *Cross-language information retrieval*, chapter 7, pages 81–90. Kluwer Academic Publishers.

Emmanuel Prochasson and Pascale Fung. 2011. Rare Word Translation Extraction from Aligned Comparable Documents. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL'11)*, pages 1327–1335, Portland, OR, USA.

Emmanuel Prochasson, Emmanuel Morin, and Kyo Kageura. 2009. Anchor points for bilingual lexicon extraction from small comparable corpora. In *Proceedings of the 12th Conference on Machine Translation Summit (MT Summit XII)*, pages 284–291, Ottawa, Canada.

Reinhard Rapp. 1995. Identify Word Translations in Non-Parallel Texts. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL'95)*, pages 320–322, Boston, MA, USA.

Reinhard Rapp. 1999. Automatic Identification of Word Translations from Unrelated English and German Corpora. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL'99)*, pages 519–526, College Park, MD, USA.

Raphaël Rubino and Georges Linarès. 2011. A multi-view approach for term translation spotting. In *Proceedings of the 12th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing'11)*, pages 29–40, Tokyo, Japan.

Kumiko Tanaka and Hideya Iwasaki. 1996. Extraction of Lexical Translations from Non-Aligned Corpora. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING'96)*, pages 580–585, Copenhagen, Denmark.

Kun Yu and Junichi Tsujii. 2009. Extracting bilingual dictionary from comparable corpora with dependency heterogeneity. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT'09)*, pages 121–124, Boulder, CO, USA.

Validating and Extending Semantic Knowledge Bases using Video Games with a Purpose

Daniele Vannella, David Jurgens, Daniele Scarfini, Domenico Toscani and **Roberto Navigli**
Department of Computer Science
Sapienza University of Rome
surname@di.uniroma1.it

Abstract

Large-scale knowledge bases are important assets in NLP. Frequently, such resources are constructed through automatic mergers of complementary resources, such as WordNet and Wikipedia. However, manually validating these resources is prohibitively expensive, even when using methods such as crowdsourcing. We propose a cost-effective method of validating and extending knowledge bases using *video games* with a purpose. Two video games were created to validate concept-concept and concept-image relations. In experiments comparing with crowdsourcing, we show that video game-based validation consistently leads to higher-quality annotations, even when players are not compensated.

1 Introduction

Large-scale knowledge bases are an essential component of many approaches in Natural Language Processing (NLP). Semantic knowledge bases such as WordNet (Fellbaum, 1998), YAGO (Suchanek et al., 2007), and BabelNet (Navigli and Ponzetto, 2010) provide ontological structure that enables a wide range of tasks, such as measuring semantic relatedness (Budanitsky and Hirst, 2006) and similarity (Pilehvar et al., 2013), paraphrasing (Kauchak and Barzilay, 2006), and word sense disambiguation (Navigli and Ponzetto, 2012; Moro et al., 2014). Furthermore, such knowledge bases are essential for building unsupervised algorithms when training data is sparse or unavailable. However, constructing and updating semantic knowledge bases is often limited by the significant time and human resources required.

Recent approaches have attempted to build or extend these knowledge bases automatically. For example, Snow et al. (2006) and Navigli (2005) extend WordNet using distributional or structural features to identify novel semantic connections between concepts. The recent advent of large semi-structured resources has enabled the creation of new semantic knowledge bases (Medelyan et al., 2009; Hovy et al., 2013) through automatically merging WordNet and Wikipedia (Suchanek et al., 2007; Navigli and Ponzetto, 2010; Niemann and Gurevych, 2011). While these automatic approaches offer the scale needed for open-domain applications, the automatic processes often introduce errors, which can prove detrimental to downstream applications. To overcome issues from fully-automatic construction methods, several works have proposed validating or extending knowledge bases using crowdsourcing (Biemann and Nygaard, 2010; Eom et al., 2012; Sarasua et al., 2012). However, these methods, too, are limited by the resources required for acquiring large numbers of responses.

In this paper, we propose validating and extending semantic knowledge bases using **video games with a purpose**. Here, the annotation tasks are transformed into elements of a video game where players accomplish their jobs by virtue of playing the game, rather than by performing a more traditional annotation task. While prior efforts in NLP have incorporated games for performing annotation and validation (Siorpaes and Hepp, 2008b; Herdağdelen and Baroni, 2012; Poesio et al., 2013), these games have largely been text-based, adding game-like features such as high-scores on top of an existing annotation task. In contrast, we introduce two video games with graphical 2D gameplay that is similar to what game players are familiar with. The fun nature of the games provides an intrinsic motivation for players to keep playing, which can increase the quality of their work and lower the cost per annotation.

Our work provides the following three contributions. First, we demonstrate effective video game-based methods for both validating and extending

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1294–1304,
Baltimore, Maryland, USA, June 23-25 2014.

semantic networks, using two games that operate on complementary sources of information: semantic relations and sense-image mappings. In contrast to previous work, the annotation quality is determined in a fully automatic way. Second, we demonstrate that converting games with a purpose into more traditional video games creates an increased player incentive such that players annotate for free, thereby significantly lowering annotation costs below that of crowdsourcing. Third, for both games, we show that games produce better quality annotations than crowdsourcing.

2 Related Work

Multiple works have proposed linguistic annotation-based games with a purpose for tasks such as anaphora resolution (Hladká et al., 2009; Poesio et al., 2013), paraphrasing (Chklovski and Gil, 2005), term associations (Artignan et al., 2009; Lafourcade and Joubert, 2010), query expansion (Simko et al., 2011), and word sense disambiguation (Chklovski and Mihalcea, 2002; Seemakurty et al., 2010; Venhuizen et al., 2013). Notably, all of these linguistic games focus on users interacting with text, in contrast to other highly successful games with a purpose in other domains, such as Foldit (Cooper et al., 2010), in which players fold protein sequences, and the ESP game (von Ahn and Dabbish, 2004), where players label images with words.

Most similar to our work are games that create or validate common sense knowledge. Two games with a purpose have incorporated video game-like mechanics for annotation. First, Herdağdelen and Baroni (2012) validate automatically acquired common sense relations using a slot machine game where players must identify valid relations and arguments from randomly aligned data within a time limit. Although the validation is embedded in a game-like setting, players are limited to one action (pulling the lever) unlike our games, which feature a variety of actions and rich gameplay experience to keep players interested longer. Second, Kuo et al. (2009) describe a pet-raising game where players must answer common sense questions in order to obtain pet food. While their game is among the most video game-like, the annotation task is a chore the player must perform in order to return to the game, rather than an integrated, fun part of the game's objectives, which potentially decreases motivation for answering correctly.

Several works have proposed adapting existing word-based board game designs to create or validate common sense knowledge. von Ahn et al. (2006) generate common sense facts by using a game similar to Taboo™, where one player must list facts about a computer-selected lemma and a second player must guess the original lemma having seen only the facts. Similarly, Vickrey et al. (2008) gather free associations to a target word with the constraint, similar to Taboo™, where players cannot enter a small set of banned words. Vickrey et al. (2008) also present two games similar to the Scattergories™, where players are given a category and then must list things in that category. The two variants differ in the constraints imposed on the players, such as beginning all items with a specific letter. For all three games, two players play the same game under time limits and then are rewarded if their answers match.

Last, three two-player games have focused on validating and extending knowledge bases. Rzeniewicz and Szymański (2013) extend WordNet with common-sense knowledge using a 20 Questions-like game. In a rapid-play style game, OntoPronto attempts to classify Wikipedia pages as either categories or individuals (Siorpaes and Hepp, 2008a). SpotTheLink uses a similar rapid question format to have players align the DBpedia and PROTON ontologies by agreeing on the distinctions between classes (Thaler et al., 2011).

Unlike dynamic gaming elements common in our video games, the above games are all focused on interacting with textual items. Another major limitation is their need for always having two players, which requires them to sustain enough interest to always maintain an active pool of players. While the computer can potentially act as a second player, such a simulated player is often limited to using preexisting knowledge or responses, which makes it difficult to validate new types of entities or create novel answers. In contrast, we drop this requirement thanks to a new strategy for assigning confidence scores to the annotations based on negative associations.

3 Video Game with a Purpose Design

To create video games, our development process focused on a common design philosophy and a common data set.

3.1 Design Objectives

Three design objectives were used to develop the video games. First, the annotation task should be a central and natural action with familiar video game mechanics. That is, the annotation should

be supplied by common actions such as collecting items, puzzles, or destroying objects, rather than through extrinsic tasks that players must complete in order to return to the game. This design has the benefits of (1) growing the annotator pool with video games players, and (2) potentially increasing annotator enjoyment.

Second, the game should be playable by a single player, with reinforcement for correct game play coming from gold standard examples.[1] We note that gold standard examples may come from both true positive and true negative items.

Third, the game design should be sufficiently general to annotate a variety of linguistic phenomena, such that only the game data need be changed to accomplish a different annotation task. While some complex linguistic annotation tasks such as preposition attachment may be difficult to integrate directly into gameplay, many simpler but still necessary annotation tasks such as word and image associations can be easily modeled with traditional video game mechanics.

3.2 Annotation Setup

Tasks We focused on two annotation tasks: (1) validating associations between two concepts, and (2) validating associations between a concept and an image. For each task we developed a video game with a purpose that integrates the task within the game, as illustrated in Sections 4 and 5.

Knowledge base As the reference knowledge base, we chose BabelNet[2] (Navigli and Ponzetto, 2010), a large-scale multilingual semantic ontology created by automatically merging WordNet with other collaboratively-constructed resources such as Wikipedia and OmegaWiki. BabelNet data offers two necessary features for generating the games' datasets. First, by connecting WordNet synsets to Wikipedia pages, most synsets are associated with a set of pictures; while often noisy, these pictures sometimes illustrate the target concept and are an ideal case for validation. Second, BabelNet contains the semantic relations from both WordNet and hyperlinks in Wikipedia; these relations are again an ideal case of validation, as not all hyperlinks connect semantically-related pages in Wikipedia. Last, we stress that while our games use BabelNet data, they could easily validate or extend other knowledge bases such as YAGO (Suchanek et al., 2007) as well.

[1] This design is in contrast to two-player games where mutual agreement reinforces correct behavior.

[2] http://babelnet.org

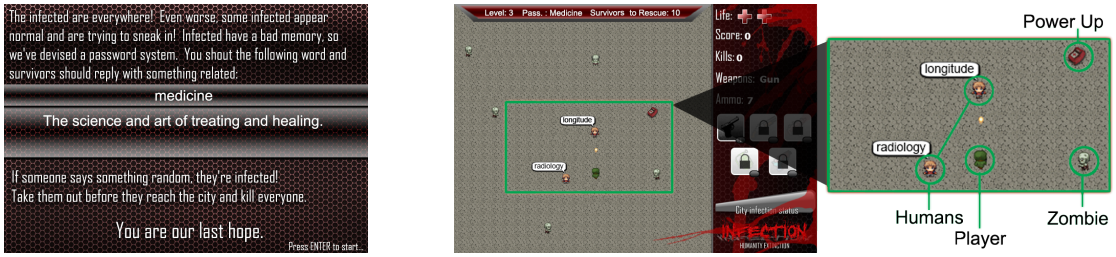
Data We created a common set of concepts, C, used in both games, containing sixty synsets selected from all BabelNet synsets with at least fifty associated images. Using the same set of synsets, separate datasets were created for the two validation tasks. In each dataset, a concept $c \in C$ is associated with two sets: a set V_c containing items to validate, and a set N_c with examples of true negative items (i.e., items where the relation to c does not hold). We use the notation V and N when referring to the to-validate and true negative sets for all concepts in a dataset, respectively.

For the concept-concept dataset, V_c is the union of V_c^B, which contains the lemmas of all synsets incident to c in BabelNet, and V_c^n, which contains novel lemmas derived from statistical associations. Specifically, novel lemmas were selected by computing the χ^2 statistic for co-occurrences between the lemmas of c and all other part of speech-tagged lemmas in Wikipedia. The 30 lemmas with the highest χ^2 are included in V_c. To enable concept-to-concept annotations, we disambiguate novel lemmas using a simple heuristic based on link co-occurrence count (Navigli and Ponzetto, 2012). Each set V_c contains 77.6 lemmas on average.

For the concept-image data, V_c is the union of V_c^B, which contains all images associated with c in BabelNet, and V_c^n, which contains web-gathered images using a lemma of c as the query. Web-gathered images were retrieved using Yahoo! Boss image search and the first result set (35 images) was added to V_c. Each set V_c contains 77.0 images on average.

For both datasets, each negative set N_c is constructed as $\cup_{c' \in C \setminus \{c\}} V_{c'}^B$, i.e., from the items related in BabelNet to all other concepts in C. By constructing N_c directly from the knowledge base, play actions may be validated based on recognition of true negatives, removing the heavy burden for ever manually creating a gold standard test set.

Annotation Aggregation In each game, an item is annotated when players make a binary choice as to whether the item's relation is true (e.g., whether an image is related to a concept). To produce a final annotation, a rating of $p - n$ is computed, where p and n denote the number of times players have marked the item's relation as true or false, respectively. Items with a positive rating after aggregating are marked as true examples of the relation and false otherwise.



(a) The passphrase shown at the start (b) Main gameplay screen with a close-up of a player's interaction with two humans

Figure 1: Screenshots of the key elements of Infection

4 Game 1: Infection

The first game, *Infection*, validates the concept-concept relation dataset.

Design Infection is designed as a top-down shooter game in the style of Commando. Infection features the classic game premise that a virus has partially infected humanity, turning people into zombies. The player's responsibility is to stop zombies from reaching the city and rescue humans that are fleeing to the city. Both zombies and humans appear at the top of the screen, advance to the bottom and, upon reaching it, enter the city.

In the game, some humans are infected, but have not yet become zombies; these infected humans must be stopped before reaching the city. Because infected and uninfected humans look identical, the player uses a passphrase call-and-response mechanism to distinguish between the two. Each level features a randomly-chosen passphrase that the player's character shouts. Uninfected humans are expected to respond with a word or phrase related to the passphrase; in contrast, infected humans have become confused due to the infection and will say something completely unrelated in an attempt to sneak past. When an infected human reaches the city, the city's total infection level increases; should the infection level increase beyond a certain threshold, the player fails the stage and must replay it to advance the game. Furthermore, if any time after ten humans have been seen, the player has killed more than 80% of the uninfected humans, the player's gun is taken by the survivors and she loses the stage.

Figure 1a shows instructions for the passphrase "medicine." In the corresponding gameplay, shown in the close up of Figure 1b, a human shouts a valid response, "radiology" for the level's passphrase, while the nearby infected human shouts an incorrect response "longitude."

Gameplay is divided into eight stages, each with increasing difficulty. Each stage has a goal of saving a specific number of uninfected humans. Infection incorporates common game mechanics, such as unlockable weapons, power-ups that restore health, and achievements. Scoring is based on both the number of zombies killed and the percentage of uninfected humans saved, motivating players to kill infected humans in order to increase their score. Importantly, Infection also includes a leaderboard where players compete for top positions based on their total scores.

Annotation Each human is assigned a response selected uniformly from V or N. Humans with responses from N are treated as infected. Players annotate by selecting which humans are infected: Allowing a human with a response from V to enter the city is treated as a positive annotation; killing that human is treated as a negative annotation.

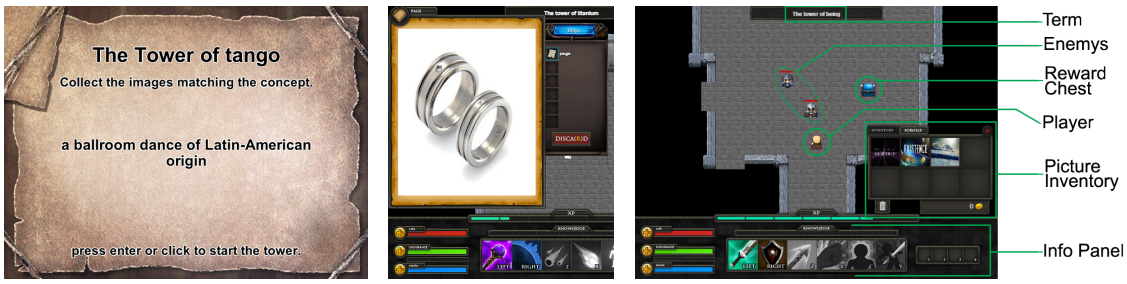
The design of Infection enables annotating multiple types of conceptual relations such as synonymy or antonymy by changing only the description of the passphrase and how uninfected humans are expected to respond.

Quality Enforcement Mechanisms Infection includes two game mechanics to limit adversarial players from creating many low quality annotations. Specifically, the game prevents players from both (1) allowing all humans to live, via the city infection level and (2) killing all humans, via survivors taking the player's gun; these actions would both generate many false positives and false negatives, respectively. These mechanics ensure the game naturally produces better quality annotations; in contrast, common crowdsourcing platforms do not support analogous mechanics for enforcing this type of correctness at annotation time.

5 Game 2: The Knowledge Towers

The second game, *The Knowledge Towers* (TKT), validates the concept-image dataset.

Design TKT is designed as a single-player role playing game (RPG) where the player explores a



(a) An example tower's concept (b) Image selection screen (c) Gameplay

Figure 2: Screenshots of the key elements of The Knowledge Towers.

series of towers to unlock long-forgotten knowledge. At the start of each tower, a target concept is shown, e.g., the tower of "tango," along with a description of the concept (Figure 2a). The player must then recover the knowledge of the target concept by acquiring pictures of it. Pictures are obtained through defeating monsters and opening treasure chests, such as those shown in Figure 2c. However, players must distinguish pictures of the tower's concept from unrelated pictures. When an image is picked up, the player may keep or discard it, as shown in Figure 2b. A player's inventory is limited to eight pictures to encourage them to select the most relevant pictures only.

Once the player has collected enough pictures, the door to the boss room is unlocked and the player may enter to defeat the boss and complete the tower. Pictures may also be deposited in special reward chests that grant experience bonuses if the deposited pictures are from V. Gathering unrelated pictures has adverse effects on the player. If the player finishes the level with a majority of unrelated pictures, the player's journey is unsuccessful and she must replay the tower.

TKT includes RPG game elements commonly found in game series such as Diablo and the Legend of Zelda: players begin with a specific character class that has class-specific skills, such as Warrior or Thief, but will unlock the ability to play as other classes by successfully completing the towers. Last, TKT includes a leaderboard where players can compete for positions; a player's score is based on increasing her character's abilities and her accuracy at discarding images from N.

Annotation Players annotate by deciding which images to keep in their inventory. Images receive positive rating annotations from: (1) depositing the image in a reward chest, and (2) ending the level with the image still in the inventory. Conversely, images receive a negative rating when a player (1) views the image but intentionally avoids picking it up or (2) drops the image from her inventory.

TKT is designed to assist in the validation and extension of automatically-created image libraries that link to semantic concepts, such as ImageNet (Deng et al., 2009) and that of Torralba et al. (2008). However, its general design allows for other types of annotations, such as image labeling, by changing the tower's instructions and pictures.

Quality Enforcement Mechanisms Similar to Infection, TKT includes analogous mechanisms for limiting adversarial player annotations. Players who collect no images are prevented from entering the boss room, limiting their ability to generate false negative annotations. Similarly, players who collect all images are likely to have half of their images from N and therefore fail the tower's quality-check after defeating the boss.

6 Experiments

Two experiments were performed with Infection and TKT: (1) an evaluation of players' ability to play accurately and to validate semantic relations and image associations and (2) a comprehensive cost comparison. Each experiment compared (a) free and financially-incentivized versions of each game, (b) crowdsourcing, and (c) a non-video game with a purpose.

6.1 Experimental Setup

Gold Standard Data To compare the quality of annotation from games and crowdsourcing, a gold standard annotation was produced for a 10% sample of each dataset (cf. Section 3.2). Two annotators independently rated the items and, in cases of disagreement, a third expert annotator adjudicated. Unlike in the game setting, annotators were free to consult additional resources such as Wikipedia.

To measure inter-annotator agreement (IAA) on the gold standard annotations, we calculated Krip-

pendorff's α (Krippendorff, 2004; Artstein and Poesio, 2008); α ranges between [-1,1] where 1 indicates complete agreement, -1 indicates systematic disagreement, and values near 0 indicate agreement at chance levels. Gold standard annotators had high agreement, 0.774, for concept-concept relations. However, image-concept agreement was only moderate, 0.549. A further analysis revealed differences in the annotators' thresholds for determining association, with one annotator permitting more abstract relations. However, the adjudication process resolved these disputes, resulting in substantial agreement by all annotators on the final gold annotations.

Incentives At the start of each game, players were shown brief descriptions of the game and a description of a contest where the top-ranked players would win either (1) monetary prizes in the form of gift cards, or (2) a mention and thanks in this paper. We refer to these as the paid and free versions of the game, respectively. In the paid setting, the five top-ranking players were offered gift cards valued at 25, 15, 15, 10, and 10 USD, starting from first place (a total of 75 USD per game). To increase competition among players and to perform a fairer time comparison with crowdsourcing, the contest period was limited to two weeks.

6.2 Comparison Methods

To compare with the video games, items were annotated using two additional methods: crowdsourcing and a non-video game with a purpose.

Crowdsourcing Setup Crowdsourcing was performed using the CrowdFlower platform. Annotation tasks were designed to closely match each game's annotation process. A task begins with a description of a target synset and its textual definition; following, ten annotation questions are shown. Separate tasks were used for validating concept-concept and concept-image relations. Each tasks' questions were shown as a binary choice of whether the item is related to the task's concept. Workers were paid 0.05 USD per task. Each question was answered by three workers.

Following common practices for guarding against adversarial workers (Mason and Suri, 2012), the tasks for concept c include quality check questions using items from N_c. Workers who rate too many relations from N_c as valid are removed by CrowdFlower and prevented from participating further. One of the ten questions in a task used an item from N_c, resulting in a task mixture of 90% annotation questions and 10% quality-check questions. However, we note that both of our video games use data that is 50% annotation, 50% quality-check. While the crowdsourcing task could be adjusted to use an increased number of quality-check options, such a design is uncommon and artificially inflates the cost of the crowdsourcing comparison beyond what would be expected. Therefore, although the crowdsourcing and game-based annotation tasks differ slightly, we chose to use the common setup in order to create a fair cost-comparison between the two.

Non-video Game with a Purpose To measure the impact of the video game itself on the annotation process, we developed a non-video game with a purpose, referred to as *SuchGame*. Players perform a single action in SuchGame: after being shown a concept c and its textual definition, a player answers whether an item is related to the concept. Items are drawn equally from V_c and N_c, with players scoring a point each time they select that an item from N is not related. A round of gameplay contains ten questions. After the round ends, players see their score for that round and the current leaderboard. Two versions of SuchGame were released, one for each dataset. SuchGame was promoted with same free recognition incentive as Infection and TKT.

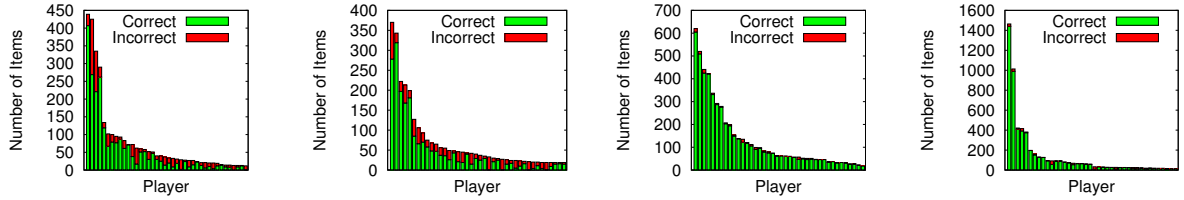
6.3 Game Release

Both video games were released to multiple online forums, social media sites, and Facebook groups. SuchGame was released to separate Facebook groups promoting free webgames and groups for indie games. For each release, we estimated an upper-bound of the audience sizes using available statistics such as Facebook group sites, website analytics, and view counts. The free and paid versions had sizes of 21,546 and 14,842 people, respectively; SuchGame had an upper bound of 569,131 people. Notices promoting the game were separated so that audiences saw promotions for one of either the paid or free incentive version. Games were also released in such a way as to preserve the anonymity of the study, which limited our ability to advertise to public venues where the anonymity might be compromised.

7 Results and Discussion

7.1 Gameplay Analysis

In this section we analyze the games in terms of participation and player's ability to correctly play. Players completed over 1388 games during the



(a) Infection (free) (b) Infection (paid) (c) TKT (free) (d) TKT (paid)

Figure 3: Accuracy of the top-40 players in rejecting true negative items during gameplay.

					G.S. Agreement			
	# Players	# Anno.	N-Acc.	Krip.'s α	True Pos.	True Neg.	All	Cost per Ann.
TKT free	100	3005	97.0	0.333	82.5	82.5	82.5	$0.000
TKT paid	97	3318	95.4	0.304	69.0	92.1	74.0	$0.023
Crowdflower	290	13854	-	0.478	59.5	93.7	66.2	$0.008
Infection free	89	3150	71.0	0.445	67.8	68.4	68.1	$0.000
Infection paid	163	3355	65.9	0.330	69.1	54.8	61.1	$0.022
Crowdflower	1097	13764	-	0.167	16.9	96.4	59.6	$0.008

Table 1: Annotation statistics from all sources. N-Accuracy denotes accuracy at rejecting items from N; G.S. Agreement denotes percentage agreement of the aggregated annotations with the gold standard.

study period. The paid and free versions of TKT had similar numbers of players, while the paid version of Infection attracted nearly twice the players compared to the free version, shown in Table 1, Column 1. However, both versions created approximately the same number of annotations, shown in Column 2. Surprisingly, SuchGame received little attention, with only a few players completing a full round of game play. We believe this emphasizes the strength of video game-based annotation; adding incentives and game-like features to an annotation task will not necessarily increase its appeal. Given SuchGame's minimal interest, we omit it from further analysis.

Second, the type of incentive did not change the percentage of items from N that players correctly reject, shown for all players as N-accuracy in Table 1 Column 3 and per-player in Figure 3. However, players were much more accurate at rejecting items from N in TKT than in Infection. We attribute this difference to the nature of the items and the format of the games. The images used by TKT provide concrete examples of a concept, which can be easily compared with the game's current concept; in addition, TKT allows players to inspect items as long as a player prefers. In contrast, concept-concept associations require more background knowledge to determine if a relation exists; furthermore, Infection gives players limited time to decide (due to board length) and also contains cognitive distractors (zombies). Nevertheless, player accuracy remains high for both games (Table 1, Col. 3) indicating the games represent a viable medium for making annotation decisions.

Last, the distribution of player annotation frequencies (Figure 3) suggests that the leaderboard and incentives motivated players. Especially in the paid condition, a clear group appears in the top five positions, which were advertised as receiving prizes. The close proximity of players in the paid positions is a result of continued competition as players jostled for higher-paying prizes.

7.2 Annotation Quality

This section assesses the annotation quality of both games and of CrowdFlower in terms of (1) the IAA of the participants, measured using Krippendorff's α, and (2) the percentage agreement of the resulting annotations with the gold standard. Players in both free and paid games had similar IAA, though the free version is consistently higher (Table 1, Col. 4).[3] For images, crowdsourcing workers have a higher IAA than game players; however, this increased agreement is due to adversarial workers consistently selecting the same, incorrect answer. In contrast, both video games contain mechanisms for limiting such behavior.

The strength of both crowdsourcing and games with a purpose comes from aggregating multiple annotations of a single item; i.e., while IAA may

[3]In conversations with players after the contest ended, several mentioned that being aware their play was contributing to research motivated them to play more accurately.

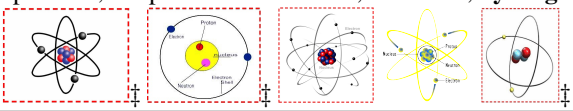
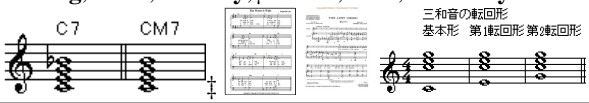
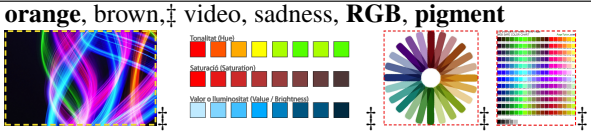
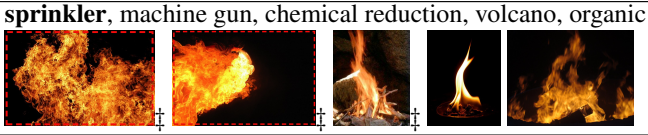
Lemma	Abbreviated Definition	Most-selected Items
atom	The smallest possible particle of a chemical element	spectrum, nonparticulate radiation, **molecule**, **hydrogen**, **electron** 
chord	A combination of three or more notes	**voicing**, **triad**, **tonality**,‡ **strum**, **note**, **harmony** 
color	An attribute from reflected or emitted light	**orange**, brown,‡ video, sadness, **RGB**, **pigment** 
fire	The state of combustion in which inflammable material burns	**sprinkler**, machine gun, chemical reduction, volcano, organic chemistry 
religion	The expression of man's belief in and reverence for a superhuman power	**polytheistic**,‡ **monotheistic**, **Jainism**, **Christianity**,‡ Freedom of religion 

Table 2: Examples of the most-selected words and images from the free version of both games. Bolded words and images with a dashed border denote items not in BabelNet. Only the items marked with a ‡ were rated as valid in the aggregated CrowdFlower annotations.

be low, the majority annotation of an item may be correct. Therefore, in Table 1, we calculate the percentage agreement of the aggregated annotations with the gold standard annotations for approving valid relations (true positives; Col. 5), rejecting invalid relations (true negatives; Col. 6), and for both combined (Col. 7). On average, both video games in all settings produce more accurate annotations than crowdsourcing. Indeed, despite having lower IAA for images, the free version of TKT provides an absolute 16.3% improvement in gold standard agreement over crowdsourcing.

Examining the difference in annotation quality for true positives and negatives, we see a strong bias with crowdsourcing towards rejecting all items. This bias leads to annotations with few false positives, but as Column 5 shows, crowdflower workers consistently performed much worse than game players at identifying valid relations, producing many false negative annotations. Indeed, for concept-concept relations, workers identified only 16.9% of the valid relations.

In contrast to crowdsourcing, both games were effective at identifying valid relations. Table 2 shows examples of the most frequently chosen items from V for the free versions of both games. For both games, players were equally likely to select novel items, suggesting the games can serve a useful purpose of adding these missing relations in automatically constructed knowledge bases. Highlighting one example, the five most selected concept-concept relations for *chord* were all novel; BabelNet included many relations to highly-specific concepts (e.g., "Circle of fifths") but did not include relations to more commonly-associated concepts, like *note* and *harmony*.

7.3 Cost Analysis

This section provides a cost-comparison between the video games and crowdsourcing. The free versions of both games proved highly successful, yielding high-quality annotations at no direct cost. Both free and paid conditions produced similar volumes of annotations, suggesting that players do not need financial incentives provided that the games are fun to play. It could be argued that the recognition incentive was motivating players in the free condition and thus some incentive was required. However, player behavior indicates otherwise: After the contest period ended, *no* players in the free setting registered for being acknowledged by name, which strongly suggests the incentive was not contributing to their motivation for playing. Furthermore, a minority of players continued to play even after the contest period ended, suggesting that enjoyment was a driving factor.

Last, while crowdsourcing has seen different quality and volume from workers in paid and unpaid settings (Rogstadius et al., 2011), in contrast, our games produced approximately-equivalent results from players in both settings.

Crowdsourcing was slightly more cost-effective than both games in the paid condition, as shown in Table 1, Column 8. However, three additional factors need to be considered. First, both games intentionally uniformly sample between V and N to increase player engagement,[4] which generates a larger number of annotations for items in N than are produced by crowdsourcing. When annotations on items in N are included for both games and crowdsourcing, the costs per annotation drop to comparable levels: \$0.007 for CrowdFlower tasks, \$0.008 for TKT, and \$0.011 for Infection.

Second, for both annotation tasks, crowdsourcing produced lower quality annotations, especially for valid relations. Based on agreement with the gold standard (Table 1, Col. 5), the estimated cost for crowdsourcing a *correct* true positive annotation increases to \$0.014 for a concept-image and a \$0.048 for concepts-concept annotation. In contrast, the cost when using video games increases only to \$0.033 for concept-image and \$0.031 for concept-concept. These cost increases suggest that crowdsourcing is not always cheaper with respect to quality.

Third, we note that both video games in the paid setting incur a fixed cost (for the prizes) and therefore additional games played can only further decrease the cost per annotation. Indeed, the present study divided the audience pool into two separate groups which effectively halved the potential number of annotations per game. Assuming combining the audiences would produce the same number of annotations, both our games' costs per annotation drop to \$0.012.

Last, video games can potentially come with indirect costs due to software development and maintenance. Indeed, Poesio et al. (2013) report spending 60,000£ in developing their Phrase Detectives game with a purpose over a two-year period. In contrast, both games here were developed as a part of student projects using open source software and assets and thus incurred no cost; furthermore, games were created in a few months, rather than years. Given that few online games attain significant sustained interest, we argue that our lightweight model is preferable for producing video games with a purpose. While using students is not always possible, the development process is fast enough to sufficiently reduce costs below those reported for Phrase Detectives.

[4]Earlier versions that used mostly items from V proved less engaging due to players frequently performing the same action, e.g., saving most humans or collecting most pictures.

8 Conclusion

Two video games have been presented for validating and extending knowledge bases. The first game, Infection, validates concept-concept relations, and the second, The Knowledge Towers, validates image-concept relations. In experiments involving online players, we demonstrate three contributions. First, games were released in two conditions whereby players either saw financial incentives for playing or a personal satisfaction incentive where they were thanked by us. We demonstrated that both conditions produced nearly identical numbers of annotations and, moreover, that players were disinterested in the satisfaction incentive, suggesting they played out of interest in the game itself. Furthermore, we demonstrated the effectiveness of a novel design for games with a purpose which does not require two players for validation and instead reinforces behavior only using true negative items that required no manual annotation. Second, in a comparison with crowdsourcing, we demonstrate that video game-based annotations consistently generated higher-quality annotations. Last, we demonstrate that video game-based annotation can be more cost-effective than crowdsourcing or annotation tasks with game-like features: The significant number of annotations generated by the satisfaction incentive condition shows that a fun game can generate high-quality annotations at virtually no cost. All annotated resources, demos of the games, and a live version of the top-ranking items for each concept are currently available online.[5]

In the future we will apply our video games to the validation of more data, such as the new Wikipedia bitaxonomy (Flati et al., 2014).

Acknowledgments

The authors gratefully acknowledge the support of the ERC Starting Grant MultiJEDI No. 259234.

We thank Francesco Cecconi for his support with the websites and the many video game players without whose enjoyment this work would not be possible.

[5]http://lcl.uniroma1.it/games/

References

Guillaume Artignan, Mountaz Hascoët, and Mathieu Lafourcade. 2009. Multiscale visual analysis of lexical networks. In *Proceedings of the International Conference on Information Visualisation*, pages 685–690.

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596.

Chris Biemann and Valerie Nygaard. 2010. Crowdsourcing wordnet. In *Proceedings of the 5th Global WordNet conference*.

Alexander Budanitsky and Graeme Hirst. 2006. Evaluating WordNet-based measures of Lexical Semantic Relatedness. *Computational Linguistics*, 32(1):13–47.

Timothy Chklovski and Yolanda Gil. 2005. Improving the design of intelligent acquisition interfaces for collecting world knowledge from web contributors. In *Proceedings of the International Conference on Knowledge Capture*, pages 35–42. ACM.

Tim Chklovski and Rada Mihalcea. 2002. Building a Sense Tagged Corpus with Open Mind Word Expert. In *Proceedings of ACL 2002 Workshop on WSD: Recent Successes and Future Directions*, Philadelphia, PA, USA.

Seth Cooper, Firas Khatib, Adrien Treuille, Janos Barbero, Jeehyung Lee, Michael Beenen, Andrew Leaver-Fay, David Baker, Zoran Popović, and Foldit players. 2010. Predicting protein structures with a multiplayer online game. *Nature*, 466(7307):756–760.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255.

Soojeong Eom, Markus Dickinson, and Graham Katz. 2012. Using semi-experts to derive judgments on word sense alignment: a pilot study. In *Proceedings of the Conference on Language Resources and Evaluation (LREC)*, pages 605–611.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Database*. MIT Press, Cambridge, MA.

Tiziano Flati, Daniele Vannella, Tommaso Pasini, and Roberto Navigli. 2014. Two is bigger (and better) than one: the Wikipedia Bitaxonomy Project. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, Baltimore, Maryland.

Amaç Herdağdelen and Marco Baroni. 2012. Bootstrapping a game with a purpose for common sense collection. *ACM Transactions on Intelligent Systems and Technology*, 3(4):1–24.

Barbora Hladká, Jiří Mírovskỳ, and Pavel Schlesinger. 2009. Play the language: Play coreference. In *Proceedings of the Joint Conference of the Association for Computational Linguistics and International Joint Conference of the Asian Federation of Natural Language Processing (ACL-IJCNLP)*, pages 209–212. Association for Computational Linguistics.

Eduard H. Hovy, Roberto Navigli, and Simone Paolo Ponzetto. 2013. Collaboratively built semi-structured content and Artificial Intelligence: The story so far. *Artificial Intelligence*, 194:2–27.

David Kauchak and Regina Barzilay. 2006. Paraphrasing for automatic evaluation. In *Proceedings of the Conference of the North American Chapter of the Association of Computational Linguistics (NAACL)*, pages 455–462.

Klaus Krippendorff. 2004. *Content Analysis: An Introduction to Its Methodology*. Sage, Thousand Oaks, CA, second edition.

Yen-ling Kuo, Jong-Chuan Lee, Kai-yang Chiang, Rex Wang, Edward Shen, Cheng-wei Chan, and Jane Yung-jen Hsu. 2009. Community-based game design: experiments on social games for commonsense data collection. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, pages 15–22.

Mathieu Lafourcade and Alain Joubert. 2010. Computing trees of named word usages from a crowdsourced lexical network. In *Proceedings of the International Multiconference on Computer Science and Information Technology (IMCSIT)*, pages 439–446, Wisla, Poland.

Winter Mason and Siddharth Suri. 2012. Conducting behavioral research on amazons mechanical turk. *Behavior Research Methods*, 44(1):1–23.

Olena Medelyan, David Milne, Catherine Legg, and Ian H. Witten. 2009. Mining meaning from Wikipedia. *International Journal of Human-Computer Studies*, 67(9):716–754.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity Linking meets Word Sense Disambiguation: A Unified Approach. *Transactions of the Association for Computational Linguistics*.

Roberto Navigli and Simone Paolo Ponzetto. 2010. BabelNet: Building a very large multilingual semantic network. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL), Uppsala, Sweden*, pages 216–225.

Roberto Navigli and Simone Paolo Ponzetto. 2012. Joining forces pays off: Multilingual Joint Word Sense Disambiguation. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1399–1410, Jeju, Korea.

Roberto Navigli. 2005. Semi-automatic extension of large-scale linguistic knowledge bases. In *Proceedings of the 18th Internationa Florida AI Research Symposium Conference,* Clearwater Beach, Florida, 15–17 May 2005, pages 548–553.

Elisabeth Niemann and Iryna Gurevych. 2011. The people's web meets linguistic knowledge: Automatic sense alignment of Wikipedia and WordNet. In *Proceedings of the International Conference on Computational Semantics (IWCS)*, pages 205–214.

Mohammad Taher Pilehvar, David Jurgens, and Roberto Navigli. 2013. Align, Disambiguate and Walk: a Unified Approach for Measuring Semantic Similarity. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1341–1351, Sofia, Bulgaria.

Massimo Poesio, Jon Chamberlain, Udo Kruschwitz, Livio Robaldo, and Luca Ducceschi. 2013. Phrase detectives: Utilizing collective intelligence for internet-scale language resource creation. *ACM Transactions on Interactive Intelligent Systems*, 3(1):3:1–3:44, April.

Jakob Rogstadius, Vassilis Kostakos, Aniket Kittur, Boris Smus, Jim Laredo, and Maja Vukovic. 2011. An assessment of intrinsic and extrinsic motivation on task performance in crowdsourcing markets. In *Proceedings of the International AAAI Conference on Weblogs and Social Media (ICWSM)*.

Jacek Rzeniewicz and Julian Szymański. 2013. Bringing Common Sense to WordNet with a Word Game. In *Computational Collective Intelligence. Technologies and Applications*, volume 8083 of *Lecture Notes in Computer Science*, pages 296–305. Springer.

Cristina Sarasua, Elena Simperl, and Natalya F Noy. 2012. CrowdMap: Crowdsourcing ontology alignment with microtasks. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 525–541.

Nitin Seemakurty, Jonathan Chu, Luis Von Ahn, and Anthony Tomasic. 2010. Word sense disambiguation via human computation. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, pages 60–63. ACM.

Jakub Simko, Michal Tvarozek, and Maria Bielikova. 2011. Little search game: term network acquisition via a human computation game. In *Proceedings of the ACM conference on Hypertext and Hypermedia*, pages 57–62.

Katharina Siorpaes and Martin Hepp. 2008a. Games with a purpose for the semantic web. *IEEE Intelligent Systems*, 23(3):50–60.

Katharina Siorpaes and Martin Hepp. 2008b. Ontogame: Weaving the semantic web by online games. In Sean Bechhofer, Manfred Hauswirth, Jrg Hoffmann, and Manolis Koubarakis, editors, *The Semantic Web: Research and Applications*, volume 5021 of *Lecture Notes in Computer Science*, pages 751–766. Springer Berlin Heidelberg.

Rion Snow, Dan Jurafsky, and Andrew Ng. 2006. Semantic taxonomy induction from heterogeneous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL), Sydney, Australia*, pages 801–808.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. YAGO: A core of semantic knowledge. unifying WordNet and Wikipedia. In *Proceedings of the 16th World Wide Web Conference,* Banff, Canada, 8–12 May 2007, pages 697–706.

Stefan Thaler, Elena Paslaru Bontas Simperl, and Katharina Siorpaes. 2011. SpotTheLink: A Game for Ontology Alignment. In *Proceedings of the 6th Conference on Professional Knowledge Management: From Knowledge to Action*, pages 246–253.

Antonio Torralba, Robert Fergus, and William T Freeman. 2008. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970.

Noortje J. Venhuizen, Valerio Basile, Kilian Evang, and Johan Bos. 2013. Gamification for word sense labeling. In *Proceedings of the International Conference on Computational Semantics (IWCS)*.

David Vickrey, Aaron Bronzan, William Choi, Aman Kumar, Jason Turner-Maier, Arthur Wang, and Daphne Koller. 2008. Online word games for semantic data collection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 533–542.

Luis von Ahn and Laura Dabbish. 2004. Labeling images with a computer game. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*, pages 319–326.

Luis von Ahn, Mihir Kedia, and Manuel Blum. 2006. Verbosity: a game for collecting common-sense facts. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*, pages 75–78.

Shallow Analysis Based Assessment of Syntactic Complexity for Automated Speech Scoring

Suma Bhat
Beckman Institute,
University of Illinois,
Urbana, IL
spbhat2@illinois.edu

Huichao Xue
Dept. of Computer Science
University of Pittsburgh
Pittsburgh, PA
hux10@cs.pitt.edu

Su-Youn Yoon
Educational Testing Service
Princeton, NJ
syoon@ets.org

Abstract

Designing measures that capture various aspects of language ability is a central task in the design of systems for automatic scoring of spontaneous speech. In this study, we address a key aspect of language proficiency assessment – syntactic complexity. We propose a novel measure of syntactic complexity for spontaneous speech that shows optimum empirical performance on real world data in multiple ways. First, it is both *robust* and *reliable*, producing automatic scores that agree well with human rating compared to the state-of-the-art. Second, the measure makes sense theoretically, both from algorithmic and native language acquisition points of view.

1 Introduction

Assessment of a speaker's proficiency in a second language is the main task in the domain of automatic evaluation of spontaneous speech (Zechner et al., 2009). Prior studies in language acquisition and second language research have conclusively shown that proficiency in a second language is characterized by several factors, some of which are, fluency in language production, pronunciation accuracy, choice of vocabulary, grammatical sophistication and accuracy. The design of automated scoring systems for non-native speaker speaking proficiency is guided by these studies in the choice of pertinent objective measures of these key aspects of language proficiency.

The focus of this study is the design and performance analysis of a measure of the syntactic complexity of non-native English responses for use in automatic scoring systems. The state-of-the art automated scoring system for spontaneous speech (Zechner et al., 2009; Higgins et al., 2011) currently uses measures of fluency and pronunciation (acoustic aspects) to produce scores that are in reasonable agreement with human-rated scores of proficiency. Despite its good performance, there is a need to extend its coverage to higher order aspects of language ability. Fluency and pronunciation may, by themselves, already be good indicators of proficiency in non-native speakers, but from a construct validity perspective[1], it is necessary that an automatic assessment model measure higher-order aspects of language proficiency. Syntactic complexity is one such aspect of proficiency. By "syntactic complexity", we mean a learner's ability to use a wide range of sophisticated grammatical structures.

This study is different from studies that focus on capturing grammatical errors in non-native speakers (Foster and Skehan, 1996; Iwashita et al., 2008). Instead of focusing on grammatical errors that are found to be highly representative of language proficiency, our interest is in capturing the *range* of forms that surface in language production and the degree of *sophistication* of such forms, collectively referred to as *syntactic complexity* in (Ortega, 2003).

The choice and design of objective measures of language proficiency is governed by two crucial constraints:

1. Validity: a measure should show high discriminative ability between various levels of language proficiency, and the scores produced by the use of this measure should show high agreement with human-assigned scores.
2. Robustness: a measures should be derived automatically and should be robust to errors in the measure generation process.

A critical impediment to the robustness constraint in the state-of-the-art is the multi-stage au-

[1]Construct validity is the degree to which a test measures what it claims, or purports, to be measuring and an important criterion in the development and use of assessments or tests.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1305–1315,
Baltimore, Maryland, USA, June 23-25 2014.

tomated process, where errors in the speech recognition stage (the very first stage) affect subsequent stages. Guided by studies in second language development, we design a measure of syntactic complexity that captures patterns indicative of proficient and non-proficient grammatical structures by a shallow-analysis of spoken language, as opposed to a deep syntactic analysis, and analyze the performance of the automatic scoring model with its inclusion. We compare and contrast the proposed measure with that found to be optimum in Yoon and Bhat (2012).

Our primary contributions in this study are:

- We show that the measure of syntactic complexity derived from a shallow-analysis of spoken utterances satisfies the design constraint of high discriminative ability between proficiency levels. In addition, including our proposed measure of syntactic complexity in an automatic scoring model results in a statistically significant performance gain over the state-of-the-art.
- The proposed measure, derived through a completely automated process, satisfies the robustness criterion reasonably well.
- In the domain of native language acquisition, the presence or absence of a grammatical structure indicates grammatical development. We observe that the proposed approach elegantly and effectively captures this presence-based criterion of grammatical development, since the feature indicative of presence or absence of a grammatical structure is optimal from an algorithmic point of view.

2 Related Work

Speaking in a non-native language requires diverse abilities, including fluency, pronunciation, intonation, grammar, vocabulary, and discourse. Informed by studies in second language acquisition and language testing that regard these factors as key determiners of spoken language proficiency, some researchers have focused on the objective measurement of these aspects of spoken language in the context of automatic assessment of language ability. Notable are studies that have focused on assessment of fluency (Cucchiarini et al., 2000; Cucchiarini et al., 2002), pronunciation (Witt and Young, 1997; Witt, 1999; Franco et al., 1997; Neumeyer et al., 2000), and intonation (Zechner et al., 2009). The relative success of these studies has yielded objective measures of acoustic aspects of speaking ability, resulting in a shift in focus to more complex aspects of assessment of grammar (Bernstein et al., 2010; Chen and Yoon, 2011; Chen and Zechner, 2011), topic development (Xie et al., 2012), and coherence (Wang et al., 2013).

In an effort to assess grammar and usage in a second language learning environment, numerous studies have focused on identifying relevant quantitative measures. These measures have been used to estimate proficiency levels in English as a second language (ESL) writing with reasonable success. Wolf-Quintero et al. (1998), Ortega (2003), and Lu (2010) found that measures such as mean length of T-unit[2] and dependent clauses per clause (henceforth termed as length-based measures) are well correlated with holistic proficiency scores suggesting that these quantitative measures can be used as objective indices of grammatical development.

In the context of spoken ESL, these measures have been studied as well but the results have been inconclusive. The measures could only broadly discriminate between students' proficiency levels, rated on a scale with moderate to weak correlations, and strong data dependencies on the participant groups were observed (Halleck, 1995; Iwashita et al., 2008; Iwashita, 2010).

With the recent interest in the area of automatic assessment of speech, there is a concurrent need to assess the grammatical development of ESL students automatically. Studies that explored the applicability of length-based measures in an automated scoring system (Chen and Zechner, 2011; Chen and Yoon, 2011) observed another important drawback of these measures in that setting. Length-based measures do not meet the constraints of the design, that, in order for measures to be effectively incorporated in the automated speech scoring system, they must be generated in a fully automated manner, via a multi-stage automated process that includes speech recognition, part of speech (POS) tagging, and parsing.

A major bottleneck in the multi-stage process of an automated speech scoring system for second language is the stage of automated speech recognition (ASR). Automatic recognition of non-native speakers' spontaneous speech is a challenging task as evidenced by the error rate of the state-of-the-

[2]T-units are defined as "the shortest grammatically allowable sentences into which writing can be split." (Hunt, 1965)

art speech recognizer. For instance, Chen and Zechner (2011) reported a 50.5% word error rate (WER) and Yoon and Bhat (2012) reported a 30% WER in the recognition of ESL students' spoken responses. These high error rates at the recognition stage negatively affect the subsequent stages of the speech scoring system in general, and in particular, during a deep syntactic analysis, which operates on a long sequence of words as its context. As a result, measures of grammatical complexity that are closely tied to a correct syntactic analysis are rendered unreliable. Not surprisingly, Chen and Zechner (2011) studied measures of grammatical complexity via syntactic parsing and found that a Pearson's correlation coefficient of 0.49 between syntactic complexity measures (derived from manual transcriptions) and proficiency scores, was drastically reduced to near non-existence when the measures were applied to ASR word hypotheses. This suggests that measures that rely on deep syntactic analysis are unreliable in current ASR-based scoring systems for spontaneous speech.

In order to avoid the problems encountered with deep analysis-based measures, Yoon and Bhat (2012) explored a shallow analysis-based approach, based on the assumption that the level of grammar sophistication at each proficiency level is reflected in the distribution of part-of-speech (POS) tag bigrams. The idea of capturing differences in POS tag distributions for classification has been explored in several previous studies. In the area of text-genre classification, POS tag distributions have been found to capture genre differences in text (Feldman et al., 2009; Marin et al., 2009); in a language testing context, it has been used in grammatical error detection and essay scoring (Chodorow and Leacock, 2000; Tetreault and Chodorow, 2008). We will see next what aspects of syntactic complexity are captured by such a shallow-analysis.

3 Shallow-analysis approach to measuring syntactic complexity

The measures of syntactic complexity in this approach are POS bigrams and are not obtained by a deep analysis (syntactic parsing) of the structure of the sentence. Hence we will refer to this approach as 'shallow analysis'. In a shallow-analysis approach to measuring syntactic complexity, we rely on the distribution of POS bigrams at every proficiency level to be representative of the range and sophistication of grammatical constructions at that level. At the outset, POS-bigrams may seem too simplistic to represent any aspect of true syntactic complexity. We illustrate to the contrary, that they are indeed able to capture certain grammatical errors and sophisticated constructions by means of the following instances. Consider the two sentence fragments below taken from actual responses (the bigrams of interest and their associated POS tags are bold-faced).

1. They ***can/MD to/TO*** survive ...
2. They created ***the culture/NN that/WDT now/RB*** is common in the US.

We notice that Example 1 is not only less grammatically sophisticated than Example 2 but also has a grammatical error. The error stems from the fact that it has a modal verb followed by the word "to". On the other hand, Example 2 contains a relative clause composed of a noun introduced by "that". Notice how these grammatical expressions (one erroneous and the other sophisticated) can be detected by the POS bigrams "MD-TO" and "NN-WDT", respectively.

The idea that the level of syntactic complexity (in terms of its range and sophistication) can be assessed based on the distribution of POS-tags is informed by prior studies in second language acquisition. It has been shown that the usage of certain grammatical constructions (such as that of the embedded relative clause in the second sentence above) are indicators of specific milestones in grammar development (Covington et al., 2006). In addition, studies such as Foster and Skehan (1996) have successfully explored the utility of frequency of grammatical errors as objective measures of grammatical development.

Based on this idea, Yoon and Bhat (2012) developed a set of features of syntactic complexity based on POS sequences extracted from a large corpus of ESL learners' spoken responses, grouped by human-assigned scores of proficiency level. Unlike previous studies, it did not rely on the occurrence of *normative* grammatical constructions. The main assumption was that each score level is characterized by different types of prominent grammatical structures. These representative constructions are gathered from a collection of ESL learners' spoken responses rated for overall proficiency. The syntactic complexity of a test spoken response was estimated based on its

similarity to the proficiency groups in the reference corpus with respect to the score-specific constructions. A score was assigned to the response based on how similar it was to the high score group. In Section 4.1, we go over the approach in further detail.

Our current work is inspired by the shallow analysis-based approach of Yoon and Bhat (2012) and operates under the same assumptions of capturing the range and sophistication of grammatical constructions at each score level. However, the approaches differ in the way in which a spoken response is assigned to a score group. We first analyze the limitations of the model studied in (Yoon and Bhat, 2012) and then describe how our model can address those limitations. The result is a new measure based on POS bigrams to assess ESL learners' mastery of syntactic complexity.

4 Models for Measuring Grammatical Competence

We mentioned that the measure proposed in this study is derived from assumptions similar to those studied in (Yoon and Bhat, 2012). Accordingly, we will summarize the previously studied model, outline its limitations, show how our proposed measure addresses those limitations and compare the two measures for the task of automatic scoring of speech.

4.1 Vector-Space Model based approach

Yoon and Bhat (2012) explored an approach inspired by information retrieval. They treat the concatenated collection of responses from a particular score-class as a 'super' document. Then, regarding POS bigrams as terms, they construct POS-based vector space models for each score-class (there are four score classes denoting levels of proficiency as will be explained in Section 5.2), thus yielding four score-specific vector-space models (VSMs). The terms of the VSM are weighted by the term frequency-inverse document frequency (tf-idf) weighting scheme (Salton et al., 1975). The intuition behind the approach is that responses in the same proficiency level often share similar grammar and usage patterns. The similarity between a test response and a score-specific vector is then calculated by a cosine similarity metric. Although a total of 4 cosine similarity scores (one per score group) were generated, only $cos_4$from among the four similarity scores, and $cosmax$, were selected as features.

- cos_4: the cosine similarity score between the test response and the vector of POS bigrams for the highest score class (level 4); and,
- $cosmax$: the score level of the VSM with which the given response shows maximum similarity.

Of these, $cos_4$was selected based on its empirical performance (it showed the strongest correlation with human-assigned scores of proficiency among the distance-based measures). In addition, an intuitive justification for the choice is that the score-4 vector is a grammatical "norm" representing the *average* grammar usage distribution of the most proficient ESL students. The measure of syntactic complexity of a response, cos_4, is its similarity to the highest score class.

The study found that the measures showed reasonable discriminative ability across proficiency levels. Despite its encouraging empirical performance, the VSM method of capturing grammatical sophistication has the following limitations.

First, the VSM-based method is likely to overestimate the contribution of the POS bigrams when highly correlated bigrams occur as terms in the VSM. Consider the presence of a grammar pattern represented by more than one POS bigram. For example, both "NN-WDT" and "WDT-RB" in Sentence 2 reflect the learner's usage of a relative clause. However, we note that the two bigrams are correlated and including them both results in an over-estimation of their contribution. The VSM set-up has no mechanism to handle correlated features.

Second, the tf-idf weighting scheme for relatively rare POS bigrams does not adequately capture their underlying distribution with respect to score groups. Grammatical expressions that occur frequently in one score level but rarely in other levels can be assumed to be characteristic of a specific score level. Therefore, the more uneven the distribution of a grammatical expression across score classes, the more important that grammatical expression should be as an indicator of a particular score class. However, the simple idf scheme cannot capture this uneven distribution. A pattern that occurs rarely but uniformly across different score groups can get the same weight as a pattern which is unevenly distributed to one score group. Martineau and Finin (2009) observed this weakness of the tf-idf weighting in the domain of sentiment

analysis. When using tf-idf weighting to extract words that were strongly associated with positive sentiment in a movie review corpus (they considered each review as a document and a word as a term), it was found that a substantial proportion of words with the highest tf-idf were rare words (e.g., proper nouns) which were not directly associated with the sentiment.

We propose to address these important limitations of the VSM approach by the use of a method that accounts for each of the deficiencies. This is done by resorting to a maximum entropy model based approach, to which we turn next.

4.2 Maximum Entropy-Based model

In order to address the limitations discussed in 4.1, we propose a classification-based approach. Taking an approach different from previous studies, we formulate the task of assigning a score of syntactic complexity to a spoken response as a classification problem: given a spoken response, assign the response to a proficiency class. A classifier is trained in an inductive fashion, using a large corpus of learner responses that is divided into proficiency scores as the training data and then used to test data that is similar to the training data. A distinguishing feature of the current study is that the measure is based on a comparison of characteristics of the test response to models trained on large amounts of data from each score point, as opposed to measures that are simply characteristics of the responses themselves (which is how measures have been considered in prior studies).

The inductive classifier we use here is the maximum-entropy model (MaxEnt) which has been used to solve several statistical natural language processing problems with much success (Berger et al., 1996; Borthwick et al., 1998; Borthwick, 1999; Pang et al., 2002; Klein et al., 2003; Rosenfeld, 2005). The productive feature engineering aspects of incorporating features into the discriminative MaxEnt classifier motivate the model choice for the problem at hand. In particular, the ability of the MaxEnt model's estimation routine to handle overlapping (correlated) features makes it directly applicable to address the first limitation of the VSM model. The second limitation, related to the ineffective weighting of terms via the the tf-idf scheme, seems to be addressed by the fact that the MaxEnt model assigns a weight to each feature (in our case, POS bigrams) on a per-class basis (in our case, score group), by taking every instance into consideration. Therefore, a MaxEnt model has an advantage over the model described in 4.1 in that it uses four different weight schemes (one per score level) and each scheme is optimized for each score level. This is beneficial in situations where the features are not evenly important across all score levels.

5 Experimental Setup

Our experiments seek answers to the following questions.

1. To what extent does a MaxEnt-score of syntactic complexity discriminate between levels of proficiency?
2. What is the effect of including the proposed measure of syntactic complexity in the state-of-the-art automatic scoring model?
3. How robust is the measure to errors in the various stages of automatic generation?

5.1 Tasks

In order to answer the motivating questions of the study, we set-up two tasks. In the first task, we compare the extent to which the VSM-based measure and the MaxEnt-based measure (outlined in 4.1 and 4.2 above) discriminate between levels of syntactic complexity. Additionally, we compare the performance of an automatic scoring model of overall proficiency that includes the measures of syntactic complexity from each of the two models being compared and analyze the gains with respect to the state-of-the-art. In the second task, we study the measures' robustness to errors incurred by ASR.

5.2 Data

In this study, we used a collection of responses from an international English language assessment. The assessment consisted of questions to which speakers were prompted to provide spontaneous spoken responses lasting approximately 45-60 seconds per question. Test takers read and/or listened to stimulus materials and then responded to questions based on the stimuli. All questions solicited spontaneous, unconstrained natural speech.

A small portion of the available data with inadequate audio quality and lack of student response was excluded from the study. The remaining responses were partitioned into two datasets: the ASR set and the scoring model training/test (SM)

set. The ASR set, with 47,227 responses, was used for ASR training and POS similarity model training. The SM set, with 2,950 responses, was used for feature evaluation and automated scoring model evaluation. There was no overlap in speakers between the ASR set and the SM set.

Each response was rated for overall proficiency by trained human scorers using a 4-point scoring scale, where 1 indicates low speaking proficiency and 4 indicated high speaking proficiency. The distribution of proficiency scores, along with other details of the data sets, are presented in Table 1.

As seen in Table 1, there is a strong bias towards the middle scores (score 2 and 3) with approximately 84-85% of the responses belonging to these two score levels. Although the skewed distribution limits the number of score-specific instances for the highest and lowest scores available for model training, we used the data without modifying the distribution since it is representative of responses in a large-scale language assessment scenario.

Human raters' extent of agreement in the subjective task of rating responses for language proficiency constrains the extent to which we can expect a machine's score to agree with that of humans. An estimate of the extent to which human raters agree on the subjective task of proficiency assessment, is obtained by two raters scoring approximately 5% of data (2,388 responses from ASR set and 140 responses from SM set). Pearson correlation r between the scores assigned by the two raters was 0.62 in ASR set and 0.58 in SM set. This level of agreement will guide the evaluation of the human-machine agreement on scores.

5.3 Stages of Automatic Grammatical Competence Assessment

Here we outline the multiple stages involved in the automatic syntactic complexity assessment. The first stage, ASR, yields an automatic transcription, which is followed by the POS tagging stage. Subsequently, the feature extraction stage (a VSM or a MaxEnt model as the case may be) generates the syntactic complexity feature which is then incorporated in a multiple linear regression model to generate a score.

The steps for automatic assessment of overall proficiency follow an analogous process (either including the POS tagger or not), depending on the objective measure being evaluated. The various objective measures are then combined in the multiple regression scoring model to generate an overall score of proficiency.

5.3.1 Automatic Speech Recognizer

An HMM recognizer was trained using ASR set (approximately 733 hours of non-native speech collected from 7,872 speakers). A gender independent triphone acoustic model and combination of bigram, trigram, and four-gram language models were used. A word error rate (WER) of 31% on the SM dataset was observed.

5.3.2 POS tagger

POS tags were generated using the POS tagger implemented in the Open-NLP toolkit[3]. It was trained on the Switchboard (SWBD) corpus. This POS tagger was trained on about 528K word/tag pairs. A combination of 36 tags from the Penn Treebank tag set and 6 tags generated for spoken languages were used in the tagger.

The tagger achieved a tagging accuracy of 96.3% on a Switchboard evaluation set composed of 379K words, suggesting high accuracy of the tagger. However, due to substantial amount of speech recognition errors in our data, the POS error rate (resulting from the combined errors of ASR and automated POS tagger) is expected to be higher.

5.3.3 VSM-based Model

We used the ASR data set to train a POS-bigram VSM for the highest score class and generated cos_4 and $cosmax$ reported in Yoon and Bhat (2012), for the SM data set as outlined in Section 4.1.

5.3.4 Maximum Entropy Model Classifier

The input to the classifier is a set of POS bigrams (1366 bigrams in all) obtained from the POS-tagged output of the data. We considered binary-valued features (whether a POS bigram occurred or not), occurrence frequency, and relative frequency as input for the purpose of experimentation. We used the maximum entropy classifier implementation in the MaxEnt toolkit[4]. The classifier was trained using the LBFGS algorithm for parameter estimation and used equal-scale gaussian priors for smoothing. The results that follow are based on MaxEnt classifier's parameter settings initialized to zero. Since a preliminary

[3] http://opennlp.apache.org

[4] `http://homepages.inf.ed.ac.uk/lzhang10/maxent_toolkit.html`.

Data set	No. of responses	No. of speakers	Score		Score distribution			
			Mean	SD	1	2	3	4
ASR	47,227	7,872	2.67	0.73	1,953 4%	16,834 36%	23,106 49%	5,334 11%
SM	2,950	500	2.61	0.74	166 6%	1,103 37%	1,385 47%	296 10%

Table 1: Data size and score distribution

analysis of the effect of varying the feature (binary or frequency) revealed that the binary-valued feature was optimal (in terms of yielding the best agreement between human and machine scores), we only report our results for this case. The ASR data set was used to train the MaxEnt classifier and the features generated from the SM data set were used for evaluation.

One straightforward way of using the maximum entropy classifier's prediction for our case is to directly use its predicted score-level – 1, 2, 3 or 4. However, this forces the classifier to make a coarse-grained choice and may over-penalize the classifier's scoring errors. To illustrate this, consider a scenario where the classifier assigns two responses A and B to score level 2 (based on the maximum *a posteriori* condition). Suppose that, for response A, the score class with the second highest probability corresponds to score level 1 and that, for response B, it corresponds to score level 3. It is apparent that the classifier has an overall tendency to assign a higher score to B, but looking at its top preference alone (2 for both responses), masks this tendency.

We thus capture the classifier's finer-grained scoring tendency by calculating the *expected value* of the classifier output. For a given response, the MaxEnt classifier calculates the conditional probability of a score-class given the response, in turn yielding conditional probabilities of each score group given the observation – p_i for score group $i \in \{1, 2, 3, 4\}$. In our case, we consider the predicted score of syntactic complexity to be the expected value of the class label given the observation as, $mescore = 1 \times p_1 + 2 \times p_2 + 3 \times p_3 + 4 \times p_4$. This permits us to better represent the score assigned by the MaxEnt classifier as a relative preference over score assignments.

5.3.5 Automatic Scoring System

We consider a multiple regression automatic scoring model as studied in Zechner et al. (2009; Chen and Zechner (2011; Higgins et al. (2011). In its state-of-the-art set-up, the following model uses the features – HMM acoustic model score (global normalized), speaking rate, word types per second, average chunk length in words and language model score (global normalized). We use these features by themselves (**Base**), and also in conjunction with the VSM-based feature (**cva4**) and the MaxEnt-based feature (**mescore**).

5.4 Evaluation Metric

We evaluate the measures using the metrics chosen in previous studies (Zechner et al., 2009; Chen and Zechner, 2011; Yoon and Bhat, 2012). A measure's utility has been evaluated according to its ability to discriminate between levels of proficiency assigned by human raters. This is done by considering the Pearson correlation coefficient between the feature and the human scores. In an ideal situation, we would have compared machine score with scores of grammatical skill assigned by human raters. In our case, however, with only access to the overall proficiency scores, we use scores of language proficiency as those of grammatical skill.

A criterion for evaluating the performance of the scoring model is the extent to which the automatic scores of overall proficiency agree with the human scores. As in prior studies, here too the level of agreement is evaluated by means of the weighted kappa measure as well as unrounded and rounded Pearson's correlations between machine and human scores (since the output of the regression model can either be rounded or regarded as is). The feature that maximizes this degree of agreement will be preferred.

6 Experimental Results

First, we compare the discriminative ability of measures of syntactic complexity (VSM-model based measure with that of the MaxEnt-based measure) across proficiency levels. Table 2 summarizes our experimental results for this task. We

Features	Manual Transcriptions	ASR
$mescore$	0.57	0.52
cos_4	0.48	0.43
$cosmax$	-	0.31

Table 2: Pearson correlation coefficients between measures and holistic proficiency scores. All values are significant at level 0.01. Only the measures cos_4 and $mescore$ were compared for robustness using manual and ASR transcriptions.

notice that of the measures compared, $mescore$ shows the highest correlation with scores of syntactic complexity. The correlation was approximately 0.1 higher in absolute value than that of cos_4, which was the best performing feature in the VSM-based model and the difference is statistically significant.

Seeking to study the robustness of the measures derived using a shallow analysis, we next compare the two measures studied here, with respect to the impact of speech recognition errors on their correlation with scores of syntactic complexity. Towards this end, we compare $mescore$ and $cos_4$when POS bigrams are extracted from manual transcriptions (ideal ASR) and ASR transcriptions.

In Table 2, noticing that the correlations decrease going along a row, we can say that the errors in the ASR system caused both $mescore$ and $cos_4$to under-perform. However, the performance drop (around 0.05) resulting from a shallow analysis is relatively small compared to the drop observed while employing a deep syntactic analysis. Chen and Zechner (2011) found that while using measures of syntactic complexity obtained from transcriptions, errors in ASR transcripts caused over 0.40 drop in correlation from that found with manual transcriptions[5]. This comparison suggests that the current POS-based shallow analysis approach is more robust to ASR errors compared to a syntactic analysis-based approach.

The effect of the measure of syntactic complexity is best studied by including it in an automatic scoring model of overall proficiency. We compare the performance gains over the state-of-the-art with the inclusion of additional features (VSM-based and MaxEnt-based, in turn). Table 3 shows the system performance with different grammar sophistication measures. The results reported are averaged over a 5-fold cross validation of the multiple regression model, where 80% of the SM data set is used to train the model and the evaluation is done using 20% of the data in every fold.

As seen in Table 3, using the proposed measure, $mescore$, leads to an improved agreement between human and machine scores of proficiency. Comparing the unrounded correlation results in Table 3 we notice that the model **Base+mescore** shows the highest correlation of predicted scores with human scores. In addition, we test the significance of the difference between two dependent correlations using Steiger's Z-test (via the `paired.r` function in the R statistical package (Revelle, 2012)). We note that the performance gain of **Base+mescore** over **Base** as well as over **Base + cos4** is statistically significant at level = 0.01. The performance gain of **Base+cos4** over **Base**, however, is not statistically significant at level = 0.01. Thus, the inclusion of the MaxEnt-based measure of syntactic complexity results in improved agreement between machine and human scores compared to the state-of-the-art model (here, **Base**).

7 Discussions

We now discuss some of the observations and results of our study with respect to the following items.

Improved performance: We sought to verify empirically that the MaxEnt model really outperforms the VSM in the case of correlated POS bigrams. To see this, we separate the test set into three subsets A, B, C. Set A contains responses where MaxEnt outperforms VSM; set B contains responses where VSM outperforms MaxEnt; set C contains responses where their predictions are comparable. For each group of responses $s \in \{A, B, C\}$, we calculate the percentage of responses P_s where two highly correlated POS bigrams occur[6]. We found that the percentages follow the order: $P_A = 12.93\% > P_C = 7.29\% >$

[5]Due to differences in the dataset and ASR system, a direct comparison between the current study and the cited prior study was not possible.

[6]We consider two POS bigrams to be highly correlated, when the their pointwise-mutual information is higher than 4.

Evaluation method	Base	Base+cos4	Base+mescore
Weighted $kappa$	0.503	0.524	**0.546**
Correlation (unrounded)	0.548	0.562	**0.592**
Correlation (rounded)	0.482	0.492	**0.519**

Table 3: Comparison of scoring model performances using features of syntactic complexity studied in this paper along with those available in the state-of-the-art. Here, **Base** is the scoring model without the measures of syntactic complexity. All correlations are significant at level 0.01.

$P_B = 4.41\%$. This suggests that when correlated POS bigrams occur, MaxEnt is more likely to provide better score predictions than VSM does.

Feature design: In the case of MaxEnt, the observation that binary-valued features (presence/absence of POS bigrams) yield better performance than features indicative of the occurrence frequency of the bigram has interesting implications. This was also observed in Pang et al. (2002) where it was interpreted to mean that overall sentiment is indicated by the presence/absence of keywords, as opposed to topic of a text, which is indicated by the repeated use of the same or similar terms. An analogous explanation is applicable here.

At first glance, the use of the presence/absence of grammatical structures may raise concerns about a potential loss of information (e.g. the distinction between an expression that is used once and another that is used multiple times is lost). However, when considered in the context of language acquisition studies, this approach seems to be justified. Studies in native language acquisition, have considered multiple grammatical developmental indices that represent the grammatical levels reached at various stages of language acquisition. For instance, Covington et al. (2006) proposed the revised D-level scale which was originally studied by Rosenberg and Abbeduto (1987). The D-Level Scale categorizes grammatical development into 8 levels according to the presence of a set of diverse grammatical expressions varying in difficulty (for example, level 0 consists of simple sentences, while level 5 consists of sentences joined by a subordinating conjunction). Similarly, Scarborough (1990) proposed the Index of Productive Syntax (IPSyn), according to which, the presence of particular grammatical structures, from a list of 60 structures (ranging from simple ones such as including only subjects and verbs, to more complex constructions such as conjoined sentences) is evidence of language acquisition milestones.

Despite the functional differences between the indices, there is a fundamental operational similarity - that they both use the presence or absence of grammatical structures, rather than their occurrence count, as evidence of acquisition of certain grammatical levels. The assumption that a presence-based view of grammatical level acquisition is also applicable to second language assessment helps validate our observation that binary-valued features yield a better performance when compared with frequency-valued features.

Generalizability: The training and test sets used in this study had similar underlying distributions – they both sought unconstrained responses to a set of items with some minor differences in item type. Looking ahead, an important question is the extent to which our measure is sensitive to a mismatch between training and test data.

8 Conclusions

Seeking alternatives to measuring syntactic complexity of spoken responses via syntactic parsers, we study a shallow-analysis based approach for use in automatic scoring.

Empirically, we show that the proposed measure, based on a maximum entropy classification, satisfied the constraints of the design of an objective measure to a high degree. In addition, the proposed measure was found to be relatively robust to ASR errors. The measure *outperformed* a related measure of syntactic complexity (also based on shallow-analysis of spoken response) previously found to be well-suited for automatic scoring. Including the measure of syntactic complexity in an automatic scoring model resulted in statistically significant performance gains over the state-of-the-art. We also make an interesting observation that the impressionistic evaluation of syntactic complexity is better approximated by the presence or absence of grammar and usage patterns (and not by their frequency of occurrence), an idea supported by studies in native language acquisition.

References

Adam L Berger, Vincent J Della Pietra, and Stephen A Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71.

Jared Bernstein, Jian Cheng, and Masanori Suzuki. 2010. Fluency and structural complexity as predictors of L2 oral proficiency. In *Proceedings of InterSpeech*, pages 1241–1244.

Andrew Borthwick, John Sterling, Eugene Agichtein, and Ralph Grishman. 1998. Exploiting diverse knowledge sources via maximum entropy in named entity recognition. In *Proc. of the Sixth Workshop on Very Large Corpora*.

Andrew Borthwick. 1999. *A maximum entropy approach to named entity recognition*. Ph.D. thesis, New York University.

Lei Chen and Su-Youn Yoon. 2011. Detecting structural events for assessing non-native speech. In *Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications*, IUNLPBEA '11, pages 38–45, Stroudsburg, PA, USA. Association for Computational Linguistics.

Miao Chen and Klaus Zechner. 2011. Computing and evaluating syntactic complexity features for automated scoring of spontaneous non-native speech. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 722–731.

Martin Chodorow and Claudia Leacock. 2000. An unsupervised method for detecting grammatical errors. In *Proceedings of NAACL*, pages 140–147.

Michael A Covington, Congzhou He, Cati Brown, Lorina Naci, and John Brown. 2006. How complex is that sentence? a proposed revision of the rosenberg and abbeduto d-level scale. *ReVision. Washington, DC http://www. ai. uga. edu/caspr/2006-01-Covington. pdf.(Accessed May 10, 2010.)*.

Catia Cucchiarini, Helmer Strik, and Lou Boves. 2000. Quantitative assessment of second language learners' fluency by means of automatic speech recognition technology. *The Journal of the Acoustical Society of America*, 107(2):989–999.

Catia Cucchiarini, Helmer Strik, and Lou Boves. 2002. Quantitative assessment of second language learners' fluency: comparisons between read and spontaneous speech. *The Journal of the Acoustical Society of America*, 111(6):2862–2873.

Sergey Feldman, M.A. Marin, Mari Ostendorf, and Maya R. Gupta. 2009. Part-of-speech histograms for genre classification of text. In *Proceedings of ICASSP*, pages 4781 –4784.

Pauline Foster and Peter Skehan. 1996. The influence of planning and task type on second language performance. *Studies in Second Language Acquisition*, 18:299–324.

Horacio Franco, Leonardo Neumeyer, Yoon Kim, and Orith Ronen. 1997. Automatic pronunciation scoring for language instruction. In *Proceedings of ICASSP*, pages 1471–1474.

Gene B Halleck. 1995. Assessing oral proficiency: a comparison of holistic and objective measures. *The Modern Language Journal*, 79(2):223–234.

Derrick Higgins, Xiaoming Xi, Klaus Zechner, and David Williamson. 2011. A three-stage approach to the automated scoring of spontaneous spoken responses. *Computer Speech & Language*, 25(2):282–306.

Kellogg W Hunt. 1965. Grammatical structures written at three grade levels. ncte research report no. 3.

Noriko Iwashita, Annie Brown, Tim McNamara, and Sally O'Hagan. 2008. Assessed levels of second language speaking proficiency: How distinct? *Applied Linguistics*, 29(1):24–49.

Noriko Iwashita. 2010. Features of oral proficiency in task performance by efl and jfl learners. In *Selected proceedings of the Second Language Research Forum*, pages 32–47.

Dan Klein, Joseph Smarr, Huy Nguyen, and Christopher D Manning. 2003. Named entity recognition with character-level models. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 180–183. Association for Computational Linguistics.

Xiaofei Lu. 2010. Automatic analysis of syntactic complexity in second language writing. *International Journal of Corpus Linguistics*, 15(4):474–496.

M.A Marin, Sergey Feldman, Mari Ostendorf, and Maya R. Gupta. 2009. Filtering web text to match target genres. In *Proceedings of ICASSP*, pages 3705–3708.

Justin Martineau and Tim Finin. 2009. Delta tfidf: An improved feature space for sentiment analysis. In *ICWSM*.

Leonardo Neumeyer, Horacio Franco, Vassilios Digalakis, and Mitchel Weintraub. 2000. Automatic scoring of pronunciation quality. *Speech Communication*, pages 88–93.

Lourdes Ortega. 2003. Syntactic complexity measures and their relationship to L2 proficiency: A research synthesis of college–level L2 writing. *Applied Linguistics*, 24(4):492–518.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

William Revelle, 2012. *psych: Procedures for Psychological, Psychometric, and Personality Research*. Northwestern University, Evanston, Illinois. R package version 1.2.1.

Sheldon Rosenberg and Leonard Abbeduto. 1987. Indicators of linguistic competence in the peer group conversational behavior of mildly retarded adults. *Applied Psycholinguistics*, 8:19–32.

Ronald Rosenfeld. 2005. *Adaptive statistical language modeling: a maximum entropy approach*. Ph.D. thesis, IBM.

Gerard Salton, Anita Wong, and Chung-Shu Yang. 1975. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

Hollis S Scarborough. 1990. Index of productive syntax. *Applied Psycholinguistics*, 11(1):1–22.

Joel R. Tetreault and Martin Chodorow. 2008. The ups and downs of preposition error detection in ESL writing. In *Proceedings of COLING*, pages 865–872.

Xinhao Wang, Keelan Evanini, and Klaus Zechner. 2013. Coherence modeling for the automated assessment of spontaneous spoken responses. In *Proceedings of NAACL-HLT*, pages 814–819.

Silke Witt and Steve Young. 1997. Performance measures for phone-level pronunciation teaching in CALL. In *Proceedings of STiLL*, pages 99–102.

Silke Witt. 1999. *Use of the speech recognition in computer-assisted language learning*. Unpublished dissertation, Cambridge University Engineering department, Cambridge, U.K.

Kate Wolf-Quintero, Shunji Inagaki, and Hae-Young Kim. 1998. Second language development in writing: Measures of fluency, accuracy, and complexity. Technical Report 17, Second Language Teaching and curriculum Center, The University of Hawai'i, Honolulu, HI.

Shasha Xie, Keelan Evanini, and Klaus Zechner. 2012. Exploring content features for automated speech scoring. In *Proceedings of the NAACL-HLT*, pages 103–111.

Su-Youn Yoon and Suma Bhat. 2012. Assessment of esl learners' syntactic competence based on similarity measures. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 600–608. Association for Computational Linguistics.

Klaus Zechner, Derrick Higgins, Xiaoming Xi, and David M Williamson. 2009. Automatic scoring of non-native spontaneous speech in tests of spoken english. *Speech Communication*, 51(10):883–895.

Can You Repeat That?
Using Word Repetition to Improve Spoken Term Detection

Jonathan Wintrode and **Sanjeev Khudanpur**
Center for Language and Speech Processing
Johns Hopkins University
jcwintr@cs.jhu.edu , khudanpur@jhu.edu

Abstract

We aim to improve *spoken term detection* performance by incorporating contextual information beyond traditional N-gram language models. Instead of taking a broad view of topic context in spoken documents, variability of word co-occurrence statistics across corpora leads us to focus instead the on phenomenon of word repetition within single documents. We show that given the detection of one instance of a term we are more likely to find additional instances of that term in the same document. We leverage this *burstiness* of keywords by taking the most confident keyword hypothesis in each document and interpolating with lower scoring hits. We then develop a principled approach to select interpolation weights using only the ASR training data. Using this re-weighting approach we demonstrate consistent improvement in the term detection performance across all five languages in the BABEL program.

1 Introduction

The *spoken term detection* task arises as a key subtask in applying NLP applications to spoken content. Tasks like topic identification and named-entity detection require transforming a continuous acoustic signal into a stream of discrete tokens which can then be handled by NLP and other statistical machine learning techniques. Given a small vocabulary of interest (1000-2000 words or multi-word terms) the aim of the term detection task is to enumerate occurrences of the keywords within a target corpus. Spoken term detection converts the raw acoustics into time-marked keyword occurrences, which may subsequently be fed (e.g. as a bag-of-terms) to standard NLP algorithms.

Although spoken term detection does not require the use of word-based automatic speech recognition (ASR), it is closely related. If we had perfectly accurate ASR in the language of the corpus, term detection is reduced to an exact string matching task. The word error rate (WER) and term detection performance are clearly correlated. Given resource constraints, domain, channel, and vocabulary limitations, particularly for languages other than English, the errorful token stream makes term detection a non-trivial task.

In order to improve detection performance, and restricting ourselves to an existing ASR system or systems at our disposal, we focus on leveraging *broad document context* around detection hypotheses. ASR systems traditionally use N-gram language models to incorporate prior knowledge of word occurrence patterns into prediction of the next word in the token stream. N-gram models cannot, however, capture complex linguistic or topical phenomena that occur outside the typical 3-5 word scope of the model. Yet, though many language models more sophisticated than N-grams have been proposed, N-grams are empirically hard to beat in terms of WER.

We consider term detection rather than the transcription task in considering how to exploit topic context, because in evaluating the retrieval of certain key terms we need not focus on improving the entire word sequence. Confidence scores from an ASR system (which incorporate N-gram probabilities) are optimized in order to produce the most likely sequence of words rather than the accuracy of individual word detections. Looking at broader document context within a more limited task might allow us to escape the limits of N-gram performance. We will show that by focusing on contextual information in the form of word repetition within documents, we obtain consistent improvement *across five languages* in the so called Base Phase of the IARPA BABEL program.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1316–1325,
Baltimore, Maryland, USA, June 23-25 2014.

1.1 Task Overview

We evaluate term detection and word repetition-based re-scoring on the IARPA BABEL training and development corpora[1] for five languages Cantonese, Pashto, Turkish, Tagalog and Vietnamese (Harper, 2011). The BABEL task is modeled on the 2006 NIST Spoken Term Detection evaluation (NIST, 2006) but focuses on limited resource conditions. We focus specifically on the so called *no target audio reuse* (NTAR) condition to make our method broadly applicable.

In order to arrive at our eventual solution, we take the BABEL Tagalog corpus and analyze word co-occurrence and repetition statistics in detail. Our observation of the variability in co-occurrence statistics between Tagalog training and development partitions leads us to narrow the scope of document context to same word co-occurrences, i.e. *word repetitions*.

We then analyze the tendency towards within-document repetition. The strength of this phenomenon suggests it may be more viable for improving term-detection than, say, topic-sensitive language models. We validate this by developing an interpolation formula to boost putative word repetitions in the search results, and then investigate a method for setting interpolation weights without manually tuning on a development set.

We then demonstrate that the method generalizes well, by applying it to the 2006 English data and the remaining four 2013 BABEL languages. We demonstrate consistent improvements in all languages in both the Full LP (80 hours of ASR training data) and Limited LP (10 hours) settings.

2 Motivation

We seek a workable definition of ***broad document context*** beyond N-gram models that will improve term detection performance on an arbitrary set of queries. Given the rise of unsupervised latent topic modeling with Latent Dirchlet Allocation (Blei et al., 2003) and similar latent variable approaches for discovering meaningful word co-occurrence patterns in large text corpora, we ought to be able to leverage these topic contexts instead of merely N-grams. Indeed there is work in the literature that shows that various topic models, latent or otherwise, can be useful for improving language model perplexity and word error rate (Khudanpur and Wu, 1999; Chen, 2009; Naptali et al., 2012). However, given the preponderance of highly frequent non-content words in the computation of a corpus' WER, it's not clear that a 1-2% improvement in WER would translate into an improvement in term detection.

Still, intuition suggests that knowing the topic context of a detected word ought to be useful in predicting whether or not a term does belong in that context. For example, if we determine the context of the detection hypothesis is about computers, containing words like 'monitor,' 'internet' and 'mouse,' then we would be more confident of a term such as 'keyboard' and less confident of a term such as 'cheese board'. The difficulty in this approach arises from the variability in word co-occurrence statistics. Using topic information will be helpful if 'monitor,' 'keyboard' and 'mouse' consistently predict that 'keyboard' is present. Unfortunately, estimates of co-occurrence from small corpora are not very consistent, and often over- or underestimate concurrence probabilities needed for term detection.

We illustrate this variability by looking at how consistent word co-occurrences are between two separate corpora in the same language: i.e., if we observe words that frequently co-occur with a keyword in the training corpus, do they also co-occur with the keywords in a second held-out corpus? Figure 1, based on the BABEL Tagalog corpus, suggests this is true only for high frequency keywords.

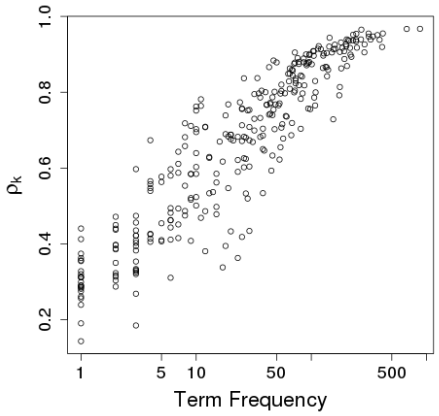
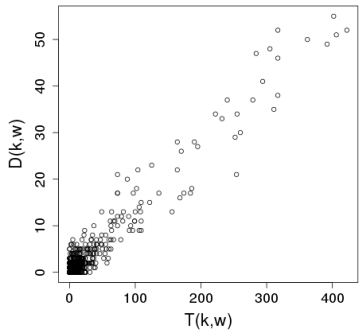


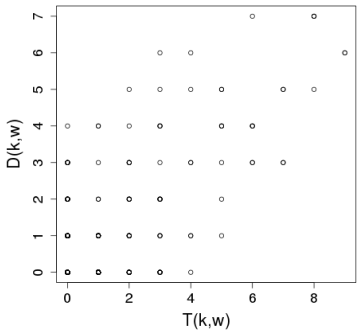
Figure 1: Correlation between the co-occurrence counts in the training and held-out sets for a fixed keyword (term) and all its "context" words.

Each point in Figure 1 represents one of 355

[1]Language collection releases IARPA-babel101-v0.4c, IARPA-babel104b-v0.4bY, IARPA-babel105b-v0.4, IARPA-babel106-v0.2g and IARPA-babel107b-v0.7 respectively.



(a) High frequency keyword 'bukas'



(b) Low frequency keyword 'Davao'

Figure 2: The number of times a fixed keyword k co-occurs with a vocabulary word w in the training speech collection — $T(k, w)$ — versus the search collection — $D(k, w)$.

Tagalog keywords used for system development by all BABEL participants. For each keyword k, we count how often it co-occurs in the same conversation as a vocabulary word w in the ASR training data and the development data, and designate the counts $T(k, w)$ and $D(k, w)$ respectively. The x-coordinate of each point in Figure 1 is the frequency of k in the training data, and the y-coordinate is the correlation coefficient ρ_k between $T(k, w)$ and $D(k, w)$. A high ρ_k implies that words w that co-occur frequently with k in the training data also do so in the search collection.

To further illustrate how Figure 1 was obtained, consider the high-frequency keyword *bukas* (count $= \mathbf{879}$) and the low-frequency keyword *Davao* (count $= \mathbf{11}$), and plot $T(k, \cdot)$ versus $D(k, \cdot)$, as done in Figure 2. The correlation coefficients ρ_{bukas} and ρ_{Davao} from the two plots end up as two points in Figure 1.

Figure 1 suggests that (k, w) co-occurrences are consistent between the two corpora ($\rho_k > 0.8$) for keywords occurring 100 or more times. However, if the goal is to help a speech retrieval system detect content-rich (and presumably infrequent) keywords, then using word co-occurrence information (i.e. topic context) does not appear to be too promising, even though intuition suggests that such information ought to be helpful.

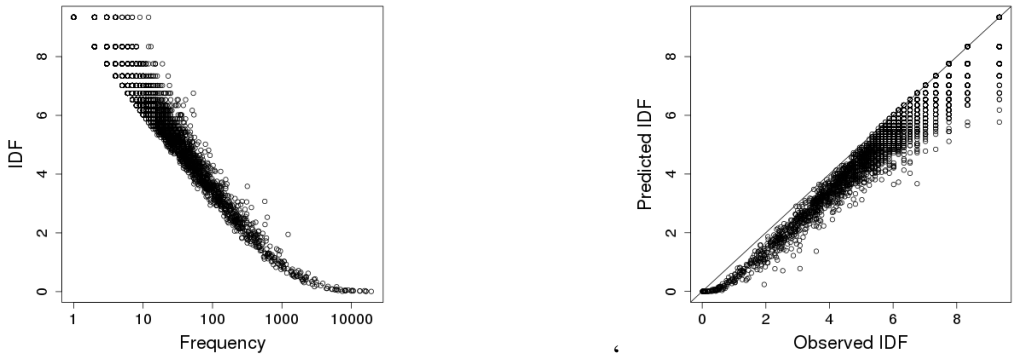
In light of this finding, we will restrict the type of ***context*** we use for term detection to the co-occurrence of the term itself elsewhere within the document. As it turns out this 'burstiness' of words within documents, as the term is defined by Church and Gale in their work on Poisson mixtures (1995), provides a more reliable framework for successfully exploiting document context.

2.1 Related Work

A number of efforts have been made to augment traditional N-gram models with latent topic information (Khudanpur and Wu, 1999; Florian and Yarowsky, 1999; Liu and Liu, 2008; Hsu and Glass, 2006; Naptali et al., 2012) including some of the early work on Probabilistic Latent Semantic Analysis by Hofmann (2001). In all of these cases WER gains in the 1-2% range were observed by interpolating latent topic information with N-gram models.

The re-scoring approach we present is closely related to adaptive or cache language models (Jelinek, 1997; Kuhn and De Mori, 1990; Kneser and Steinbiss, 1993). The primary difference between this and previous work on similar language models is the narrower focus here on the term detection task, in which we consider each search term in isolation, rather than all words in the vocabulary. Most recently, Chiu and Rudnicky (2013) looked at word bursts in the IARPA BABEL conversational corpora, and were also able to successfully improve performance by leveraging the burstiness of language. One advantage of the approach proposed here, relative to their approach, is its simplicity and its not requiring an additional tuning set to estimate parameters.

In the information retrieval community, clustering and latent topic models have yielded improvements over traditional vector space models. We will discuss in detail in the following section related works by Church and Gale (1995, 1999, and 2000). Work by Wei and Croft (2006) and Chen (2009) take a language model-based approach to



(a) f_w versus IDF_w

(b) Obsered versus predicted IDF_w

Figure 3: Tagalog corpus frequency statistics, unigrams

information retrieval, and again, interpolate latent topic models with N-grams to improve retrieval performance. However, in many text retrieval tasks, queries are often tens or hundreds of words in length rather than short spoken phrases. In these efforts, the topic model information was helpful in boosting retrieval performance above the baseline vector space or N-gram models.

Clearly topic or context information is relevant to a retrieval type task, but we need a stable, consistent framework in which to apply it.

3 Term and Document Frequency Statistics

To this point we have assumed an implicit property of low-frequency words which Church and Gale state concisely in their 1999 study of *inverse document frequency*:

> Low frequency words tend to be rich in content, and vice versa. But not all equally frequent words are equally meaningful. Church and Gale (1999).

The typical use of Document Frequency (DF) in information retrieval or text categorization is to emphasize words that occur in only a few documents and are thus more "rich in content". Close examination of DF statistics by Church and Gale in their work on Poisson Mixtures (1995) resulted in an analysis of the *burstiness* of content words.

In this section we look at DF and *burstiness* statistics applying some of the analyses of Church and Gale (1999) to the BABEL Tagalog corpus. We observe, in 648 Tagalog conversations, similar phenomena as observed by Church and Gale on 89,000 AP English newswire articles. We proceed in this fashion to make a case for why burstiness ought to help in the term detection task.

For the Tagalog conversations, as with English newswire, we observe that the document frequency, DF_w, of a word w is not a linear function of word frequency f_w in the log domain, as would be expected under a naive Poisson generative assumption. The implication of deviations from a Poisson model is that *words tend to be concentrated in a small number of documents* rather than occurring uniformly across the corpus. This is the *burstiness* we leverage to improve term detection.

The first illustration of word burstiness can be seen by plotting observed inverse document frequency, IDF_w, versus f_w in the log domain (Figure 3a). We use the same definition of IDF_w as Church and Gale (1999):

$$\mathrm{IDF}_w = -\log_2 \frac{\mathrm{DF}_w}{N}, \tag{1}$$

where N is the number of documents (i.e. conversations) in the corpus.

There is good linear correlation ($\rho = 0.73$) between $\log f_w$ and IDF_w. Yet, visually, the relationship in Figure 3a is clearly not linear. In contrast, the AP English data exhibits a correlation of $\rho = 0.93$ (Church and Gale, 1999). Thus the deviation in the Tagalog corpus is more pronounced, i.e. words are less uniformly distributed across documents.

A second perspective on word burstiness that follows from Church and Gale (1999) is that a Poisson assumption should lead us to predict:

$$\widehat{\mathrm{IDF}}_w = -\log_2 \left(1 - e^{-\frac{f_w}{N}}\right). \tag{2}$$

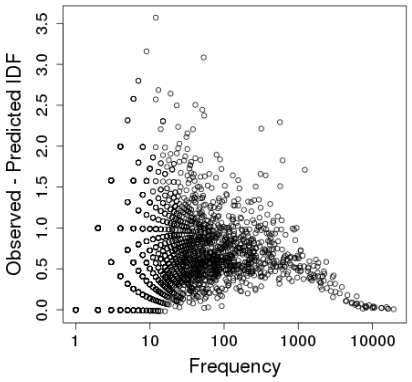


Figure 4: Difference between observed and predicted IDF_w for Tagalog unigrams.

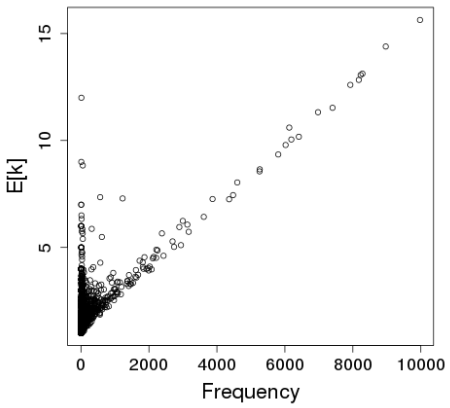


Figure 5: Tagalog *burstiness*.

For the AP newswire, Church and Gale found the largest deviation between the predicted $\widehat{\mathrm{IDF}}_w$ and observed IDF_w to occur in the middle of the frequency range. We see a somewhat different picture for Tagalog speech in Figure 3b. Observed IDF_w values again deviate significantly from their predictions (2), but all along the frequency range.

There is a noticeable quantization effect occurring in the high IDF range, given that our N is at least a factor of 100 smaller than the number of AP articles they studied: 648 vs. 89,000. Figure 4 also shows the difference between and observed IDF_w and Poisson estimate $\widehat{\mathrm{IDF}}_w$ and further illustrates the high variance in IDF_w for low frequency words.

Two questions arise: what is happening with infrequent words, and why does this matter for term detection? To look at the data from a different perspective, we consider the random variable k, which is the number of times a word occurs in a particular document. In Figure 5 we plot the following ratio, which Church and Gale (1995) define as *burstiness* :

$$E_w[k|k > 0] = \frac{f_w}{\mathrm{DF}_w} \tag{3}$$

as a function of f_w. We denote this as $E[k]$ and can interpret burstiness as the expected word count given we see w at least once.

In Figure 5 we see two classes of words emerge. A similar phenomenon is observed concerning adaptive language models (Church, 2000). In general, we can think of using word repetitions to re-score term detection as applying a limited form of adaptive or cache language model (Jelinek, 1997). Likewise, Katz attempts to capture these two classes in his G model of word frequencies (1996).

For the first class, burstiness increases slowly but steadily as w occurs more frequently. Let us label these Class A words. Since our corpus size is fixed, we might expect this to occur, as more word occurrences must be pigeon-holed into the same number of documents

Looking close to the y-axis in Figure 5, we observe a second class of exclusively low frequency words whose burstiness ranges from highly concentrated to singletons. We will refer to these as Class B words. If we take the Class A concentration trend as typical, we can argue that most Class B words exhibit a larger than average concentration. In either case we see evidence that *both high and low frequency words tend towards repeating within a document*.

3.1 Unigram Probabilities

In applying the *burstiness* quantity to term detection, we recall that the task requires us to locate a particular instance of a term, not estimate a count, hence the utility of N-gram language models predicting words in sequence.

We encounter the burstiness property of words again by looking at unigram occurrence probabilities. We compare the unconditional unigram probability (the probability that a given word token is w) with the conditional unigram probability, *given the term has occurred once in the document*. We compute the conditional probability for w using frequency information.

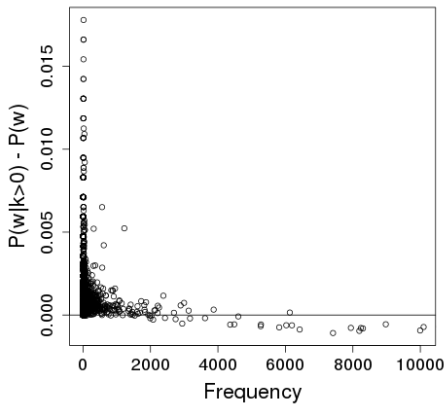


Figure 6: Difference between conditional and unconditional unigram probabilities for Tagalog

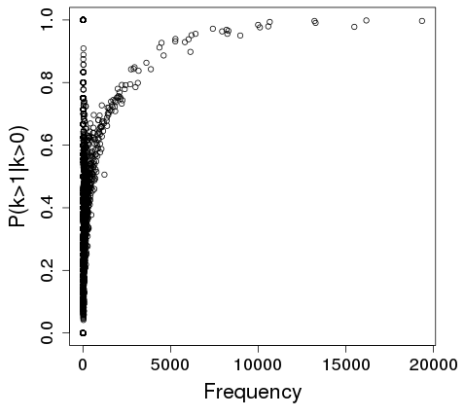


Figure 7: Tagalog word adaptation probability

$$P(w|k>0) = \frac{f_w - \mathrm{DF}_w}{\sum_{D:w\in D}|D|} \quad (4)$$

Figure 6 shows the difference between conditional and unconditional unigram probabilities. Without any other information, Zipf's law suggests that most word types do not occur in a particular document. However, conditioning on one occurrence, most word types are more likely to occur again, due to their burstiness.

Finally we measure the *adaptation* of a word, which is defined by Church and Gale (1995) as:

$$P_{adapt}(w) = P_w(k>1|k>0) \quad (5)$$

When we plot adaptation versus f_w (Figure 7) we see that all high-frequency and a significant number of low-frequency terms have adaptation greater that 50%. To be precise, 26% of all tokens and 25% of low-frequency ($f_w < 100$) have at least 50% adaptation. Given that adaptation values are roughly an order of magnitude higher than the conditional unigram probabilities, in the next two sections we describe how we use adaptation to boost term detection scores.

4 Term Detection Re-scoring

We summarize our re-scoring of repeated words with the observation: *given a correct detection, the likelihood of additional terms in the same documents should increase*. When we observe a term detection score with high confidence, we boost the other lower-scoring terms in the same document to reflect this increased likelihood of repeated terms.

For each term t and document d we propose interpolating the ASR confidence score for a particular detection t_d with the top scoring hit in d which we'll call $\widehat{t}_d$.

$$S(t_d) = (1-\alpha)P_{\mathrm{asr}}(t_d|O) + \alpha P_{\mathrm{asr}}(\widehat{t}_d|O) \quad (6)$$

We will we develop a principled approach to selecting α using the adaptation property of the corpus. However to verify that this approach is worth pursuing, we sweep a range of small α values, on the assumption that we still do want to mostly rely on the ASR confidence score for term detection. For the Tagalog data, we let α range from 0 (the baseline) to 0.4 and re-score each term detection score according to (6). Table 1 shows the results of this parameter sweep and yields us 1 to 2% absolute performance gains in a number of term detection metrics.

α	**ATWV**	$P(\mathrm{Miss})$
0.00	0.470	0.430
0.05	0.481	0.422
0.10	0.483	0.420
0.15	0.484	0.418
0.20	0.483	0.416
0.25	0.480	0.417
0.30	0.477	0.417
0.35	0.475	0.415
0.40	0.471	0.413
0.45	0.465	0.413
0.50	0.462	0.410

Table 1: Term detection scores for swept α values on Tagalog development data

The primary metric for the BABEL program, Actual Term Weighted Value (ATWV) is defined by NIST using a cost function of the false alarm probability $P(\text{FA})$ and $P(\text{Miss})$, averaged over a set of queries (NIST, 2006). The manner in which the components of ATWV are defined:

$$P(\text{Miss}) = 1 - N_{\text{true}}(\text{term})/f_{\text{term}} \quad (7)$$

$$P(\text{FA}) = N_{\text{false}}/Duration_{\text{corpus}} \quad (8)$$

implies that cost of a miss is inversely proportional to the frequency of the term in the corpus, but the cost of a false alarm is fixed. For this reason, we report both ATWV and the $P(\text{Miss})$ component. A decrease in $P(\text{Miss})$ reflects the fact that we are able to boost correct detections of the repeated terms.

4.1 Interpolation Weights

We would prefer to use prior knowledge rather than naive tuning to select an interpolation weight α. Our analysis of word burstiness suggests that *adaptation*, is a reasonable candidate. Adaptation also has the desirable property that we can estimate it for each word in the training vocabulary directly from training data and not post-hoc on a per-query basis. We consider several different estimates and we can show that the favorable result extends across languages.

Intuition suggests that we prefer per-term interpolation weights related to the term's *adaptation*. But despite the strong evidence of the adaptation phenomenon in both high and low-frequency words (Figure 7), we have less confidence in the adaptation strength of any particular word.

As with word co-occurrence, we consider if estimates of $P_{adapt}(w)$ from training data are consistent when estimated on development data. Figure 8 shows the difference between $P_{adapt}(w)$ measured on the two corpora (for words occurring in both).

We see that the adaptation estimates are only consistent between corpora for high-frequency words. Using this $P_{adapt}(w)$ estimate directly actually hurts ATWV performance by 4.7% absolute on the 355 term development query set (Table 2).

Given the variability in estimating $P_{adapt}(w)$, an alternative approach would be take $\widehat{P_w}$ as an upper bound on α, reached as the DF_w increases (cf. Equation 9). We would discount the adaptation factor when DF_w is low and we are unsure of the effect.

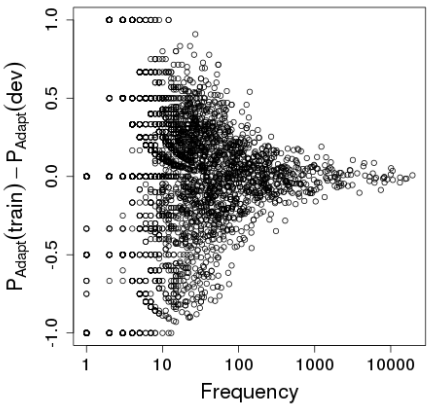


Figure 8: Difference in adaptation estimates between Tagalog training and development corpora

Interpolation Weight	**ATWV**	$P(\text{Miss})$
None	0.470	0.430
$P_{adapt}(w)$	0.423	0.474
$(1 - e^{-\text{DF}_w})P_{adapt}(w)$	0.477	0.415
$\widehat{\alpha}$ = 0.20	**0.483**	**0.416**

Table 2: Term detection performance using various interpolation weight strategies on Tagalog dev data

$$\alpha_w = (1 - e^{-\text{DF}_w}) \cdot \widehat{P}_{adapt}(w) \quad (9)$$

This approach shows a significant improvement (0.7% absolute) over the baseline. However, considering this estimate in light of the two classes of words in Figure 5, there are clearly words in Class B with high burstiness that will be ignored by trying to compensate for the high adaptation variability in the low-frequency range.

Alternatively, we take a weighted average of α_w's estimated on training transcripts to obtain a single $\widehat{\alpha}$ per language (cf. Equation 10).

$$\widehat{\alpha} = \underset{w}{\text{Avg}} \left[\left(1 - e^{-\text{DF}_w}\right) \cdot \widehat{P}_{adapt}(w) \right] \quad (10)$$

Using this average as a single interpolation weight for all terms gives near the best performance as we observed in our parameter sweep. Table 2 contrasts the results for using the three different interpolation heuristics on the Tagalog development queries. Using the mean $\widehat{\alpha}$ instead of individual α_w's provides an additional 0.5% absolute

Language	$\widehat{\alpha}$	**ATWV (%±)**	$P(\text{Miss})$ **(%±)**
Full LP setting			
Tagalog	0.20	**0.523** (+1.1)	0.396 (-1.9)
Cantonese	0.23	**0.418** (+1.3)	0.458 (-1.9)
Pashto	0.19	**0.419** (+1.1)	0.453 (-1.6)
Turkish	0.14	**0.466** (+0.8)	0.430 (-1.3)
Vietnamese	0.30	**0.420** (+0.7)	0.445 (-1.0)
English (Dev06)	0.20	**0.670 (+0.3)**	0.240 (-0.4)
Limited LP setting			
Tagalog	0.22	**0.228** (+0.9)	0.692 (-1.7)
Cantonese	0.26	**0.205** (+1.0)	0.684 (-1.3)
Pashto	0.21	**0.206** (+0.9)	0.682 (-0.9)
Turkish	0.16	**0.202** (+1.1)	0.700 (-0.8)
Vietnamese	0.34	**0.227** (+1.0)	0.646 (+0.4)

Table 3: Word-repetition re-scored results for available CTS term detection corpora

improvement, suggesting that we find additional gains boosting low-frequency words.

5 Results

Now that we have tested word repetition-based re-scoring on a small Tagalog development set we want to know if our approach, and particularly our $\widehat{\alpha}$ estimate is sufficiently robust to apply broadly. At our disposal, we have the five BABEL languages — Tagalog, Cantonese, Pashto, Turkish and Vietnamese — as well as the development data from the NIST 2006 English evaluation. The BABEL evaluation query sets contain roughly 2000 terms each and the 2006 English query set contains roughly 1000 terms.

The procedure we follow for each language condition is as follows. We first estimate adaptation probabilities from the ASR training transcripts. From these we take the weighted average as described previously to obtain a single interpolation weight $\widehat{\alpha}$ for each training condition. We train ASR acoustic and language models from the training corpus using the Kaldi speech recognition toolkit (Povey et al., 2011) following the default BABEL training and search recipe which is described in detail by Chen et al. (2013). Lastly, we re-score the search output by interpolating the top term detection score for a document with subsequent hits according to Equation 6 using the $\widehat{\alpha}$ estimated for this training condition.

For each of the BABEL languages we consider both the FullLP (80 hours) and LimitedLP (10 hours) training conditions. For the English system, we also train a Kaldi system on the 240 hours of the Switchboard conversational English corpus. Although Kaldi can produce multiple types of acoustic models, for simplicity we report results using discriminatively trained Subspace Gaussian Mixture Model (SGMM) acoustic output densities, but we do find that similar results can be obtained with other acoustic model configurations.

Using our final algorithm, we are able to boost repeated term detections and improve results in **all languages and training conditions**. Table 3 lists complete results and the associated estimates for $\widehat{\alpha}$. For the BABEL languages, we observe improvements in ATWV from 0.7% to 1.3% absolute and reductions in the miss rate of 0.8% to 1.9%. The only test for which $P(\text{Miss})$ did not improve was the Vietnamese Limited LP setting, although overall ATWV did improve, reflecting a lower $P(\text{FA})$.

In all conditions we also obtain α estimates which correspond to our expectations for particular languages. For example, adaptation is lowest for the agglutinative Turkish language where longer word tokens should be less likely to repeat. For Vietnamese, with shorter, syllable length word tokens, we observe the lowest adaptation estimates.

Lastly, the reductions in $P(\text{Miss})$ suggests that we are improving the term detection metric, which is sensitive to threshold changes, by doing what we set out to do, which is to boost lower confidence repeated words and correctly asserting them

as true hits. Moreover, we are able to accomplish this in a wide variety of languages.

6 Conclusions

Leveraging the **burstiness** of content words, we have developed a simple technique to consistently boost term detection performance across languages. Using word repetitions, we effectively use a broad document context outside of the typical 2-5 N-gram window. Furthermore, we see improvements across a broad spectrum of languages: languages with syllable-based word tokens (Vietnamese, Cantonese), complex morphology (Turkish), and dialect variability (Pashto).

Secondly, our results are not only effective but also intuitive, given that the interpolation weight parameter matches our expectations for the burstiness of the word tokens in the language on which it is estimated.

We have focused primarily on re-scoring results for the term detection task. Given the effectiveness of the technique across multiple languages, we hope to extend our effort to exploit our human tendency towards redundancy to decoding or other aspects of the spoken document processing pipeline.

Acknowledgements

This work was partially supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense U.S. Army Research Laboratory (DoD / ARL) contract number W911NF-12-C-0015. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government.

Insightful discussions with Chiu and Rudnicky (2013) are also gratefully acknowledged.

References

David Blei, Andrew Ng, and Michael Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

Guoguo Chen, Sanjeev Khudanpur, Daniel Povey, Jan Trmal, David Yarowsky, and Oguz Yilmaz. 2013. Quantifying the value of pronunciation lexicons for keyword search in low resource languages. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE.

Berlin Chen. 2009. Latent topic modelling of word co-occurence information for spoken document retrieval. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3961–3964. IEEE.

Justin Chiu and Alexander Rudnicky. 2013. Using conversational word bursts in spoken term detection. In *Proceedings of the 14th Annual Conference of the International Speech Communication Association*, pages 2247–2251. ISCA.

Kenneth Church and William Gale. 1995. Poisson Mixtures. *Natural Language Engineering*, 1(2):163–190.

Kenneth Church and William Gale. 1999. Inverse Focument Frequency (IDF): A measure of deviations from Poisson. In *Natural Language Processing Using Very Large Corpora*, pages 283–295. Springer.

Kenneth Church. 2000. Empirical estimates of adaptation: the chance of two Noriegas is closer to p/2 than p 2. In *Proceedings of the 18th Conference on Computational Linguistics*, volume 1, pages 180–186. ACL.

Radu Florian and David Yarowsky. 1999. Dynamic nonlocal language modeling via hierarchical topic-based adaptation. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*, pages 167–174. ACL.

Mary Harper. 2011. IARPA Solicitation IARPA-BAA-11-02. `http://www.iarpa.gov/solicitations_babel.html`.

Thomas Hofmann. 2001. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1):177–196.

Bo-June Paul Hsu and James Glass. 2006. Style & topic language model adaptation using HMM-LDA. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*. ACL.

Fred Jelinek. 1997. *Statistical Methods for Speech Recognition*. MIT Press.

Slava Katz. 1996. Distribution of content words and phrases in text and language modelling. *Natural Language Engineering*, 2(1):15–59.

Sanjeev Khudanpur and Jun Wu. 1999. A maximum entropy language model integrating n-grams and topic dependencies for conversational speech recognition. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 553–556. IEEE.

Reinhard Kneser and Volker Steinbiss. 1993. On the dynamic adaptation of stochastic language models. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages 586–589. IEEE.

Roland Kuhn and Renato De Mori. 1990. A cache-based natural language model for speech recognition. *Transactions on Pattern Analysis and Machine Intelligence*, 12(6):570–583.

Yang Liu and Feifan Liu. 2008. Unsupervised language model adaptation via topic modeling based on named entity hypotheses. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing, (ICASSP)*, pages 4921–4924. IEEE.

Welly Naptali, Masatoshi Tsuchiya, and Seiichi Nakagawa. 2012. Topic-dependent-class-based n-gram language model. *Transactions on Audio, Speech, and Language Processing*, 20(5):1513–1525.

NIST. 2006. The Spoken Term Detection (STD) 2006 Evaluation Plan. `http://www.itl.nist.gov/iad/mig/tests/std/2006/docs/std06-evalplan-v10.pdf`. [Online; accessed 28-Feb-2013].

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. 2011. The Kaldi speech recognition toolkit. In *Proceedings of the Automatic Speech Recognition and Understanding Workshop (ASRU)*.

Xing Wei and W Bruce Croft. 2006. LDA-based document models for ad-hoc retrieval. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 178–185. ACM.

Character-Level Chinese Dependency Parsing

Meishan Zhang[†], **Yue Zhang**[‡] , **Wanxiang Che**[†], **Ting Liu**[†]*
[†]Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
{mszhang, car, tliu}@ir.hit.edu.cn
[‡]Singapore University of Technology and Design
yue_zhang@sutd.edu.sg

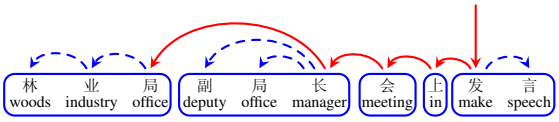
Abstract

Recent work on Chinese analysis has led to large-scale annotations of the internal structures of words, enabling character-level analysis of Chinese syntactic structures. In this paper, we investigate the problem of character-level Chinese dependency parsing, building dependency trees over characters. Character-level information can benefit downstream applications by offering flexible granularities for word segmentation while improving word-level dependency parsing accuracies. We present novel adaptations of two major shift-reduce dependency parsing algorithms to character-level parsing. Experimental results on the Chinese Treebank demonstrate improved performances over word-based parsing methods.

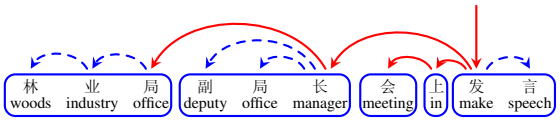
1 Introduction

As a light-weight formalism offering syntactic information to downstream applications such as SMT, the dependency grammar has received increasing interest in the syntax parsing community (McDonald et al., 2005; Nivre and Nilsson, 2005; Carreras et al., 2006; Duan et al., 2007; Koo and Collins, 2010; Zhang and Clark, 2008; Nivre, 2008; Bohnet, 2010; Zhang and Nivre, 2011; Choi and McCallum, 2013). Chinese dependency trees were conventionally defined over *words* (Chang et al., 2009; Li et al., 2012), requiring word segmentation and POS-tagging as pre-processing steps. Recent work on Chinese analysis has embarked on investigating the syntactic roles of characters, leading to large-scale annotations of word internal structures (Li, 2011; Zhang et al., 2013). Such annotations enable dependency parsing on the character level, building dependency trees over Chinese *characters*. Figure 1(c) shows an example of a character-level dependency tree, where the leaf nodes are Chinese characters.

(a) a word-based dependency tree



(b) a character-level dependency tree by Zhao (2009) with real intra-word and pseudo inter-word dependencies



(c) a character-level dependency tree investigated in this paper with both real intra- and inter-word dependencies

Figure 1: An example character-level dependency tree. "林业局副局长在大会上发言 (The deputy director of forestry administration make a speech in the meeting)".

Character-level dependency parsing is interesting in at least two aspects. First, character-level trees circumvent the issue that no universal standard exists for Chinese word segmentation. In the well-known Chinese word segmentation bakeoff tasks, for example, different segmentation standards have been used by different data sets (Emerson, 2005). On the other hand, most disagreement on segmentation standards boils down to disagreement on segmentation granularity. As demonstrated by Zhao (2009), one can extract both fine-grained and coarse-grained words from character-level dependency trees, and hence can adapt to flexible segmentation standards using this formalism. In Figure 1(c), for example, "副局长 (deputy

*Corresponding author.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1326–1336,
Baltimore, Maryland, USA, June 23-25 2014.

director)" can be segmented as both "副 (deputy) | 局长 (director)" and "副局长 (deputy director)", but not "副 (deputy) 局 (office) | 长 (manager)", by dependency coherence. Chinese language processing tasks, such as machine translation, can benefit from flexible segmentation standards (Zhang et al., 2008; Chang et al., 2008).

Second, word internal structures can also be useful for syntactic parsing. Zhang et al. (2013) have shown the usefulness of word structures in Chinese constituent parsing. Their results on the Chinese Treebank (CTB) showed that character-level constituent parsing can bring increased performances even with the pseudo word structures. They further showed that better performances can be achieved when manually annotated word structures are used instead of pseudo structures.

In this paper, we make an investigation of character-level Chinese dependency parsing using Zhang et al. (2013)'s annotations and based on a transition-based parsing framework (Zhang and Clark, 2011). There are two dominant transition-based dependency parsing systems, namely the arc-standard and the arc-eager parsers (Nivre, 2008). We study both algorithms for character-level dependency parsing in order to make a comprehensive investigation. For direct comparison with word-based parsers, we incorporate the traditional word segmentation, POS-tagging and dependency parsing stages in our joint parsing models. We make changes to the original transition systems, and arrive at two novel transition-based character-level parsers.

We conduct experiments on three data sets, including CTB 5.0, CTB 6.0 and CTB 7.0. Experimental results show that the character-level dependency parsing models outperform the word-based methods on all the data sets. Moreover, manually annotated intra-word dependencies can give improved word-level dependency accuracies than pseudo intra-word dependencies. These results confirm the usefulness of character-level syntax for Chinese analysis. The source codes are freely available at `http://sourceforge.net/projects/zpar/, version 0.7`.

2 Character-Level Dependency Tree

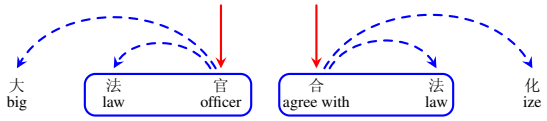
Character-level dependencies were first proposed by Zhao (2009). They show that by annotating character dependencies within words, one can adapt to different segmentation standards. The dependencies they study are restricted to intra-word characters, as illustrated in Figure 1(b). For inter-word dependencies, they use a pseudo right-headed representation.

In this study, we integrate inter-word syntactic dependencies and intra-word dependencies using large-scale annotations of word internal structures by Zhang et al. (2013), and study their interactions. We extract unlabeled dependencies from bracketed word structures according to Zhang et al.'s head annotations. In Figure 1(c), the dependencies shown by dashed arcs are *intra-word* dependencies, which reflect the internal word structures, while the dependencies with solid arcs are *inter-word* dependencies, which reflect the syntactic structures between words.

In this formulation, a character-level dependency tree satisfies the same constraints as the traditional word-based dependency tree for Chinese, including projectivity. We differentiate intra-word dependencies and inter-word dependencies by the arc type, so that our work can be compared with conventional word segmentation, POS-tagging and dependency parsing pipelines under a canonical segmentation standard.

The character-level dependency trees hold to a specific word segmentation standard, but are not limited to it. We can extract finer-grained words of different granulities from a coarse-grained word by taking projective subtrees of different sizes. For example, taking all the intra-word modifier nodes of "长 (manager)" in Figure 1(c) results in the word "副局长 (deputy director)", while taking the first modifier node of "长 (manager)" results in the word "局长 (director)". Note that "副局 (deputy office)" cannot be a word because it does not form a projective span without "长 (manager)".

Inner-word dependencies can also bring benefits to parsing word-level dependencies. The head character can be a less sparse feature compared to a word. As intra-word dependencies lead to fine-grained subwords, we can also use these subwords for better parsing. In this work, we use the innermost left/right subwords as atomic features. To extract the subwords, we find the innermost left/right modifiers of the head character, respectively, and then conjoin them with all their descendant characters to form the smallest left/right subwords. Figure 2 shows an example, where the smallest left subword of "大法官 (chief lawyer)" is "法官 (lawyer)", and the smallest right subword



(a) smallest left subword (b) smallest right subword

Figure 2: An example to illustrate the innermost left/right subwords.

of "合法化 (legalize)" is "合法 (legal)".

3 Character-Level Dependency Parsing

A transition-based framework with global learning and beam search decoding (Zhang and Clark, 2011) has been applied to a number of natural language processing tasks, including word segmentation, POS-tagging and syntactic parsing (Zhang and Clark, 2010; Huang and Sagae, 2010; Bohnet and Nivre, 2012; Zhang et al., 2013). It models a task incrementally from a start state to an end state, where each intermediate state during decoding can be regarded as a partial output. A number of actions are defined so that the state advances step by step. To learn the model parameters, it usually uses the online perceptron algorithm with early-update under the inexact decoding condition (Collins, 2002; Collins and Roark, 2004). Transition-based dependency parsing can be modeled under this framework, where the state consists of a stack and a queue, and the set of actions can be either the arc-eager (Zhang and Clark, 2008) or the arc-standard (Huang et al., 2009) transition systems.

When the internal structures of words are annotated, character-level dependency parsing can be treated as a special case of word-level dependency parsing, with "words" being "characters". A big weakness of this approach is that full words and POS-tags cannot be used for feature engineering. Both are crucial to well-established features for word segmentation, POS-tagging and syntactic parsing. In this section, we introduce novel extensions to the arc-standard and the arc-eager transition systems, so that word-based and character-based features can be used simultaneously for character-level dependency parsing.

3.1 The Arc-Standard Model

The arc-standard model has been applied to joint segmentation, POS-tagging and dependency parsing (Hatori et al., 2012), but with pseudo word structures. For unified processing of annotated word structures and fair comparison between character-level arc-eager and arc-standard systems, we define a different arc-standard transition system, consistent with our character-level arc-eager system.

In the word-based arc-standard model, the transition state includes a stack and a queue, where the stack contains a sequence of partially-parsed dependency trees, and the queue consists of unprocessed input words. Four actions are defined for state transition, including *arc-left* (AL, which creates a left arc between the top element s_0 and the second top element s_1 on the stack), *arc-right* (AR, which creates a right arc between s_0 and s_1), *pop-root* (PR, which defines the root node of a dependency tree when there is only one element on the stack and no element in the queue), and the last *shift* (SH, which shifts the first element q_0 of the queue onto the stack).

For character-level dependency parsing, there are two types of dependencies: inter-word dependencies and intra-word dependencies. To parse them with both character and word features, we extend the original transition actions into two categories, for inter-word dependencies and intra-word dependencies, respectively. The actions for inter-word dependencies include *inter-word arc-left* (AL_w), *inter-word arc-right* (AR_w), *pop-root* (PR) and *inter-word shift* (SH_w). Their definitions are the same as the word-based model, with one exception that the *inter-word shift* operation has a parameter denoting the POS-tag of the incoming word, so that POS disambiguation is performed by the SH_w action.

The actions for intra-word dependencies include *intra-word arc-left* (AL_c), *intra-word arc-right* (AR_c), *pop-word* (PW) and *inter-word shift* (SH_c). The definitions of AL_c, AR_c and SH_c are the same as the word-based arc-standard model, while PW changes the top element on the stack into a full-word node, which can only take inter-word dependencies. One thing to note is that, due to variable word sizes in character-level parsing, the number of actions can vary between different sequences of actions corresponding to different analyses. We use the padding method (Zhu et al., 2013), adding an *IDLE* action to finished transition action sequences, for better alignments between states in the beam.

In the character-level arc-standard transition

step	action	stack	queue	dependencies
0	-	ϕ	林 业 $\cdots$	ϕ
1	SH_w(NR)	林/NR	业 局 $\cdots$	ϕ
2	SH_c	林/NR 业/NR	局 副 $\cdots$	ϕ
3	AL_c	业/NR	局 副 $\cdots$	$A_1 = \{林^\frown业\}$
4	SH_c	业/NR 局/NR	副 局 $\cdots$	A_1
5	AL_c	局/NR	副 局 $\cdots$	$A_2 = A_1 \bigcup \{业^\frown局\}$
6	PW	林业局/NR	副 局 $\cdots$	A_2
7	SH_w(NN)	林业局/NR 副/NN	局 长 $\cdots$	A_2
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
12	PW	林业局/NR 副局长/NN	会 上 $\cdots$	A_i
13	AL_w	副局长/NN	会 上 $\cdots$	$A_{i+1} = A_i \bigcup \{林业局/NR^\frown副局长/NN\}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$

(a) character-level dependency parsing using the arc-standard algorithm

step	action	stack	deque	queue	dependencies
0	-	ϕ		林 业 $\cdots$	
1	SH_c(NR)	ϕ	林/NR	业 局 $\cdots$	ϕ
2	AL_c	ϕ	ϕ	业/NR 局 $\cdots$	$A_1 = \{林^\frown业\}$
3	SH_c	ϕ	业/NR	局 副 $\cdots$	A_1
4	AL_c	ϕ	ϕ	局/NR 副 $\cdots$	$A_2 = A_1 \bigcup \{业^\frown局\}$
5	SH_c	ϕ	局/NR	副 局 $\cdots$	A_2
6	PW	ϕ	林业局/NR	副 局 $\cdots$	A_2
7	SH_w	林业局/NR	ϕ	副 局 $\cdots$	A_2
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
13	PW	林业局/NR	副局长/NN	会 上 $\cdots$	A_i
14	AL_w	ϕ	副局长/NN	会 上 $\cdots$	$A_{i+1} = A_i \bigcup \{林业局/NR^\frown副局长/NN\}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$

(b) character-level dependency parsing using the arc-eager algorithm, $t = 1$

Figure 3: Character-level dependency parsing of the sentence in Figure 1(c).

system, each word is initialized by the action SH_w with a POS tag, before being incrementally modified by a sequence of intra-word actions, and finally being completed by the action PW. The inter-word actions can be applied when all the elements on the stack are full-word nodes, while the intra-word actions can be applied when at least the top element on the stack is a partial-word node. For the actions AL_c and AR_c to be valid, the top two elements on the stack are both partial-word nodes. For the action PW to be valid, only the top element on the stack is a partial-word node. Figure 3(a) gives an example action sequence.

There are three types of features. The first two types are traditionally established features for the dependency parsing and joint word segmentation and POS-tagging tasks. We use the features proposed by Hatori et al. (2012). The word-level dependency parsing features are added when the inter-word actions are applied, and the features for joint word segmentation and POS-tagging are added when the actions PW, SH_w and SH_c are applied. Following the work of Hatori et al. (2012), we have a parameter α to adjust the weights for joint word segmentation and POS-tagging features. We apply word-based dependency parsing features to intra-word dependency parsing as well, by using subwords (the conjunction of characters spanning the head node) to replace words in word features. The third type of features is word-structure features. We extract the head character and the smallest subwords containing the head character from the intra-word dependencies (Section 2). Table 1 summarizes the features.

3.2 The Arc-Eager Model

Similar to the arc-standard case, the state of a word-based arc-eager model consists of a stack and a queue, where the stack contains a sequence of partial dependency trees, and the queue consists of unprocessed input words. Unlike the arc-standard model, which builds dependencies on the top two elements on the stack, the arc-eager model builds dependencies between the top element of the stack and the first element of the queue. Five actions are defined for state transformation: *arc-left* (AL, which creates a left arc between the top element of the stack s_0 and the first element in the queue q_0, while popping s_0 off the stack), *arc-right* (AR, which creates a right arc between

Feature templates
$L\underline{c}$, $L\underline{c}\underline{t}$, $R\underline{c}$, $R\underline{c}\underline{t}$, $L_{lc1}\underline{c}$, $L_{rc1}\underline{c}$, $R_{lc1}\underline{c}$,
$L\underline{c} \cdot R\underline{c}$, $L_{lc1}\underline{c}\underline{t}$, $L_{rc1}\underline{c}\underline{t}$, $R_{lc1}\underline{c}\underline{t}$,
$L\underline{c} \cdot R\underline{w}$, $L\underline{w} \cdot R\underline{c}$, $L\underline{c}\underline{t} \cdot R\underline{w}$,
$L\underline{w}\underline{t} \cdot R\underline{c}$, $L\underline{w} \cdot R\underline{c}\underline{t}$, $L\underline{c} \cdot R\underline{w}\underline{t}$,
$L\underline{c} \cdot R\underline{c} \cdot L_{lc1}\underline{c}$, $L\underline{c} \cdot R\underline{c} \cdot L_{rc1}\underline{c}$,
$L\underline{c} \cdot R\underline{c} \cdot L_{lc2}\underline{c}$, $L\underline{c} \cdot R\underline{c} \cdot L_{rc2}\underline{c}$,
$L\underline{c} \cdot R\underline{c} \cdot R_{lc1}\underline{c}$, $L\underline{c} \cdot R\underline{c} \cdot R_{lc2}\underline{c}$,
$Ll\underline{sw}$, $Lr\underline{sw}$, $Rl\underline{sw}$, $Rr\underline{sw}$, $Ll\underline{swt}$,
$Lr\underline{swt}$, $Rl\underline{swt}$, $Rr\underline{swt}$, $Ll\underline{sw} \cdot R\underline{w}$,
$Lr\underline{sw} \cdot R\underline{w}$, $L\underline{w} \cdot Rl\underline{sw}$, $L\underline{w} \cdot Rr\underline{sw}$

Table 1: Feature templates encoding intra-word dependencies. L and R denote the two elements over which the dependencies are built; the subscripts $lc1$ and $rc1$ denote the left-most and right-most children, respectively; the subscripts $lc2$ and $rc2$ denote the second left-most and second right-most children, respectively; $\underline{w}$ denotes the word; $\underline{t}$ denotes the POS tag; $\underline{c}$ denotes the head character; $\underline{lsw}$ and $\underline{rsw}$ denote the smallest left and right subwords respectively, as shown in Figure 2.

s_0 and q_0, while shifting q_0 from the queue onto the stack), *pop-root* (PR, which defines the ROOT node of the dependency tree when there is only one element on the stack and no element in the queue), *reduce* (RD, which pops s_0 off the stack), and *shift* (SH, which shifts q_0 onto the stack).

There is no previous work that exploits the arc-eager algorithm for jointly performing POS-tagging and dependency parsing. Since the first element of the queue can be shifted onto the stack by either SH or AR, it is more difficult to assign a POS tag to each word by using a single action. In this work, we make a change to the configuration state, adding a deque between the stack and the queue to save partial words with intra-word dependencies. We divide the transition actions into two categories, one for inter-word dependencies (AR_w, AL_w, SH_w, RD_w and PR) and the other for intra-word dependencies (AR_c, AL_c, SH_c, RD_c and PW), requiring that the intra-word actions be operated between the deque and the queue, while the inter-word actions be operated between the stack and the deque.

For character-level arc-eager dependency parsing, the inter-word actions are the same as the word-based methods. The actions AL_c and AR_c are the same as AL_w and AR_w, except that they operate on characters, but the SH_c operation has a parameter to denote the POS tag of a word. The PW action recognizes a full-word. We also have an *IDLE* action, for the same reason as the arc-standard model.

In the character-level arc-eager transition system, a word is formed in a similar way with that of character-level arc-standard algorithm. Each word is initialized by the action SH_c with a POS tag, and then incrementally changed a sequence of intra-word actions, before being finalized by the action PW. All these actions operate between the queue and deque. For the action PW, only the first element in the deque (close to the queue) is a partial-word node. For the actions AR_c and AL_c to be valid, the first element in the deque must be a partial-word node. The action SH_c have a POS tag when shifting the first character of a word,but does not have such a parameter when shifting the next characters of a word. For the action SH_c with a POS tag to be valid, the first element in the deque must be a full-word node. Different from the arc-standard model, at any stage we can choose either the action SH_c with a POS tag to initialize a new word on the deque, or the inter-word actions on the stack. In order to eliminate the ambiguity, we define a new parameter t to limit the max size of the deque. If the deque is full with t words, inter-word actions are performed; otherwise intra-word actions are performed. All the inter-word actions must be applied on full-word nodes between the stack an the deque. Figure 3(b) gives an example action sequence.

Similar to the arc-standard case, there are three types of features, with the first two types being traditionally established features for dependency parsing and joint word segmentation and POS-tagging. The dependency parsing features are taken from the work of Zhang and Nivre (2011), and the features for joint word segmentation and POS-tagging are taken from Zhang and Clark (2010)[1]. The word-level dependency parsing features are triggered when the inter-word actions are applied, while the features of joint word segmentation and POS-tagging are added when the actions SH_c, AR_c and PW are applied. Again we use a parameter α to adjust the weights for joint word segmentation and POS-tagging features. The word-level features for dependency parsing are applied to intra-word dependency parsing as well, by using subwords to replace words. The third type of features is word-structure features, which are the

[1]Since Hatori et al. (2012) also use Zhang and Clark (2010)'s features, the arc-standard and arc-eager character-level dependency parsing models have the same features for joint word segmentation and POS-tagging.

		CTB50	CTB60	CTB70
Training	#sent	18k	23k	31k
	#word	494k	641k	718k
Development	#sent	350	2.1k	10k
	#word	6.8k	60k	237k
	#oov	553	3.3k	13k
Test	#sent	348	2.8k	10k
	#word	8.0k	82k	245k
	#oov	278	4.6k	13k

Table 2: Statistics of datasets.

same as those of the character-level arc-standard model, shown in Table 1.

4 Experiments

4.1 Experimental Settings

We use the Chinese Penn Treebank 5.0, 6.0 and 7.0 to conduct the experiments, splitting the corpora into training, development and test sets according to previous work. Three different splitting methods are used, namely CTB50 by Zhang and Clark (2010), CTB60 by the official documentation of CTB 6.0, and CTB70 by Wang et al. (2011). The dataset statistics are shown in Table 2. We use the head rules of Zhang and Clark (2008) to convert phrase structures into dependency structures. The intra-word dependencies are extracted from the annotations of Zhang et al. (2013)[2].

The standard measures of word-level precision, recall and F1 score are used to evaluate word segmentation, POS-tagging and dependency parsing, following Hatori et al. (2012). In addition, we use the same measures to evaluate intra-word dependencies, which indicate the performance of predicting word structures. A word's structure is correct only if all the intra-word dependencies are all correctly recognized.

4.2 Baseline and Proposed Models

For the baseline, we have two different pipeline models. The first consists of a joint segmentation and POS-tagging model (Zhang and Clark, 2010) and a word-based dependency parsing model using the arc-standard algorithm (Huang et al., 2009). We name this model STD (pipe). The second consists of the same joint segmentation and POS-tagging model and a word-based dependency parsing model using the arc-eager algorithm (Zhang and Nivre, 2011). We name this model EAG (pipe). For the pipeline models, we use a beam of size 16 for joint segmentation and POS-tagging, and a beam of size 64 for dependency parsing, according to previous work.

We study the following character-level dependency parsing models:

- STD (real, pseudo): the arc-standard model with annotated intra-word dependencies and pseudo inter-word dependencies;
- STD (pseudo, real): the arc-standard model with pseudo intra-word dependencies and real inter-word dependencies;
- STD (real, real): the arc-standard model with annotated intra-word dependencies and real inter-word dependencies;
- EAG (real, pseudo): the arc-eager model with annotated intra-word dependencies and pseudo inter-word dependencies;
- EAG (pseudo, real): the arc-eager model with pseudo intra-word dependencies and real inter-word dependencies;
- EAG (real, real): the arc-eager model with annotated intra-word dependencies and real inter-word dependencies.

The annotated intra-word dependencies refer to the dependencies extracted from annotated word structures, while the pseudo intra-word dependencies used in the above models are similar to those of Hatori et al. (2012). For a given word $w = c_1 c_2 \cdots c_m$, the intra-word dependency structure is $c_1 \curvearrowleft c_2 \curvearrowleft \cdots \curvearrowleft c_m$[3]. The real inter-word dependencies refer to the syntactic word-level dependencies by head-finding rules from CTB, while the pseudo inter-word dependencies refer to the word-level dependencies used by Zhao (2009) ($w_1 \curvearrowleft w_2 \curvearrowleft \cdots \curvearrowleft w_n$). The character-level models with annotated intra-word dependencies and pseudo inter-word dependencies are compared with the pipelines on word segmentation and POS-tagging accuracies, and are compared with the character-level models with annotated intra-word dependencies and real inter-word dependencies on word segmentation, POS-tagging and word-structure predicating accuracies. All the proposed

[2] https://github.com/zhangmeishan/wordstructures; their annotation was conducted on CTB 5.0, while we made annotations of the remainder of the CTB 7.0 words. We also make the annotations publicly available at the same site.

[3] We also tried similar structures with right arcs, which gave lower accuracies.

STD (real, real)	SEG	POS	DEP	WS
$\alpha = 1$	95.85	91.60	76.96	95.14
$\alpha = 2$	96.09	91.89	77.28	**95.29**
$\alpha = 3$	96.02	91.84	77.22	95.23
$\alpha = \mathbf{4}$	**96.10**	**91.96**	**77.49**	**95.29**
$\alpha = 5$	96.07	91.90	77.31	95.21

Table 3: Development test results of the character-level arc-standard model on CTB60.

EAG (real, real)		SEG	POS	DEP	WS
$\alpha = 1$	$t = 1$	**96.00**	91.66	74.63	**95.49**
	$t = 2$	95.93	91.75	76.60	95.37
	$\mathbf{t = 3}$	95.93	**91.74**	**76.94**	95.36
	$t = 4$	95.91	91.71	76.82	95.33
	$t = 5$	95.95	91.73	76.84	95.40
$t = 3$	$\alpha = 1$	95.93	91.74	76.94	95.36
	$\alpha = 2$	96.11	91.99	77.17	95.56
	$\alpha = \mathbf{3}$	**96.16**	**92.01**	**77.48**	**95.62**
	$\alpha = 4$	96.11	91.93	77.40	95.53
	$\alpha = 5$	96.00	91.84	77.10	95.43

Table 4: Development test results of the character-level arc-eager model on CTB60.

models use a beam of size 64 after considering both speeds and accuracies.

4.3 Development Results

Our development tests are designed for two purposes: adjusting the parameters for the two proposed character-level models and testing the effectiveness of the novel word-structure features. Tuning is conducted by maximizing word-level dependency accuracies. All the tests are conducted on the CTB60 data set.

4.3.1 Parameter Tuning

For the arc-standard model, there is only one parameter α that needs tuning. It adjusts the weights of segmentation and POS-tagging features, because the number of feature templates is much less for the two tasks than for parsing. We set the value of α to $1 \cdots 5$, respectively. Table 3 shows the accuracies on the CTB60 development set. According to the results, we use $\alpha = 4$ for our final character-level arc-standard model.

For the arc-eager model, there are two parameters t and α. t denotes the deque size of the arc-eager model, while α shares the same meaning as the arc-standard model. We take two steps for parameter tuning, first adjusting the more crucial parameter t and then adjusting α on the best t. Both parameters are assigned the values of 1 to 5. Table 4 shows the results. According to results, we set $t = 3$ and $\alpha = 3$ for the final character-level arc-eager model, respectively.

	SEG	POS	DEP	WS
STD (real, real)	96.10	91.96	77.49	95.29
STD (real, real)/wo	95.99	91.79	77.19	95.35
Δ	-0.11	-0.17	-0.30	+0.06
EAG (real, real)	96.16	92.01	77.48	95.62
EAG (real, real)/wo	96.09	91.82	77.12	95.56
Δ	-0.07	-0.19	-0.36	-0.06

Table 5: Feature ablation tests for the novel word-structure features, where "/wo" denotes the corresponding models without the novel intra-word dependency features.

4.3.2 Effectiveness of Word-Structure Features

To test the effectiveness of our novel word-structure features, we conduct feature ablation experiments on the CTB60 development data set for the proposed arc-standard and arc-eager models, respectively. Table 5 shows the results. We can see that both the two models achieve better accuracies on word-level dependencies with the novel word-structure features, while the features do not affect word-structure predication significantly.

4.4 Final Results

Table 6 shows the final results on the CTB50, CTB60 and CTB70 data sets, respectively. The results demonstrate that the character-level dependency parsing models are significantly better than the corresponding word-based pipeline models, for both the arc-standard and arc-eager systems. Similar to the findings of Zhang et al. (2013), we find that the annotated word structures can give better accuracies than pseudo word structures. Another interesting finding is that, although the arc-eager algorithm achieves lower accuracies in the word-based pipeline models, it obtains comparative accuracies in the character-level models.

We also compare our results to those of Hatori et al. (2012), which is comparable to STD (pseudo, real) since similar arc-standard algorithms and features are used. The major difference is the set of transition actions. We rerun their system on the three datasets[4]. As shown in Table 6, our arc-standard system with pseudo word structures

[4] `http://triplet.cc/`. We use a different constituent-to-dependency conversion scheme in comparison with Hatori et al. (2012)'s work.

Model	CTB50				CTB60				CTB70			
	SEG	POS	DEP	WS	SEG	POS	DEP	WS	SEG	POS	DEP	WS
The arc-standard models												
STD (pipe)	97.53	93.28	79.72	–	95.32	90.65	75.35	–	95.23	89.92	73.93	–
STD (real, pseudo)	97.78	93.74	–	**97.40**	**95.77**‡	91.24‡	–	**95.08**	**95.59**‡	90.49‡	–	**94.97**
STD (pseudo, real)	97.67	94.28‡	81.63‡	–	95.63‡	**91.40**‡	76.75‡	–	95.53‡	90.75‡	75.63‡	–
STD (real, real)	**97.84**	**94.62**‡	**82.14**‡	97.30	95.56‡	91.39‡	**77.09**‡	94.80	95.51‡	**90.76**‡	**75.70**‡	94.78
Hatori+ '12	97.75	94.33	81.56	–	95.26	91.06	75.93	–	95.27	90.53	74.73	–
The arc-eager models												
EAG (pipe)	97.53	93.28	79.59	–	95.32	90.65	74.98	–	95.23	89.92	73.46	–
EAG (real, pseudo)	97.75	93.88	–	97.45	95.63‡	91.07‡	–	95.06	**95.50**‡	90.36‡	–	**95.00**
EAG (pseudo, real)	97.76	**94.36**‡	81.70‡	–	95.63‡	91.34‡	76.87‡	–	95.39‡	90.56‡	75.56‡	–
EAG (real, real)	**97.84**	**94.36**‡	**82.07**‡	**97.49**	**95.71**‡	**91.51**‡	**76.99**‡	**95.16**	95.47‡	**90.72**‡	**75.76**‡	94.94

Table 6: Main results, where the results marked with ‡ denote that the p-value is less than 0.001 compared with the pipeline word-based models using pairwise t-test.

brings consistent better accuracies than their work on all the three data sets.

Both the pipelines and character-level models with pseudo inter-word dependencies perform word segmentation and POS-tagging jointly, without using real word-level syntactic information. A comparison between them (STD/EAG (pipe) vs. STD/EAG (real, pseudo)) reflects the effectiveness of annotated intra-word dependencies on segmentation and POS-tagging. We can see that both the arc-standard and arc-eager models with annotated intra-word dependencies can improve the segmentation accuracies by 0.3% and the POS-tagging accuracies by 0.5% on average on the three datasets. Similarly, a comparison between the character-level models with pseudo inter-word dependencies and the character-level models with real inter-word dependencies (STD/EAG (real, pseudo) vs. STD/EAG (real, real)) can reflect the effectiveness of annotated inter-word structures on morphology analysis. We can see that improved POS-tagging accuracies are achieved using the real inter-word dependencies when jointly performing inner- and inter-word dependencies. However, we find that the inter-word dependencies do not help the word-structure accuracies.

4.5 Analysis

To better understand the character-level parsing models, we conduct error analysis in this section. All the experiments are conducted on the CTB60 test data sets. The new advantage of the character-level models is that one can parse the internal word structures of intra-word dependencies. Thus we are interested in their capabilities of predicting word structures. We study the word-structure accuracies in two aspects, including OOV, word length, POS tags and the parsing model.

4.5.1 OOV

The word-structure accuracy of OOV words reflects a model's ability of handling unknown words. The overall recalls of OOV word structures are 67.98% by STD (real, real) and 69.01% by EAG (real, real), respectively. We find that most errors are caused by failures of word segmentation. We further investigate the accuracies when words are correctly segmented, where the accuracies of OOV word structures are 87.64% by STD (real, real) and 89.07% by EAG (real, real). The results demonstrate that the structures of Chinese words are not difficult to predict, and confirm the fact that Chinese word structures have some common syntactic patterns.

4.5.2 Parsing Model

From the above analysis in terms of OOV, word lengths and POS tags, we can see that the EAG (real, real) model and the STD (real, real) models behave similarly on word-structure accuracies. Here we study the two models more carefully, comparing their word accuracies sentence by sentence. Figure 4 shows the results, where each point denotes a sentential comparison between STD (real, real) and EAG (real, real), the x-axis denotes the sentential word-structure accuracy of STD (real, real), and the y-axis denotes that of EAG (real, real). The points at the diagonal show the same accuracies by the two models, while others show that the two models perform differently on the corresponding sentences. We can see that most points are beyond the diagonal line, indicat-

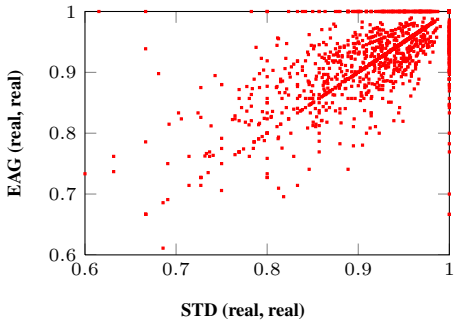


Figure 4: Sentential word-structure accuracies of STD (real, real) and EAG (real, real).

ing that the two parsing models can be complementary in parsing intra-word dependencies.

5 Related Work

Zhao (2009) was the first to study character-level dependencies; they argue that since no consistent word boundaries exist over Chinese word segmentation, dependency-based representations of word structures serve as a good alternative for Chinese word segmentation. Thus their main concern is to parse intra-word dependencies. In this work, we extend their formulation, making use of large-scale annotations of Zhang et al. (2013), so that the syntactic word-level dependencies can be parsed together with intra-word dependencies.

Hatori et al. (2012) proposed a joint model for Chinese word segmentation, POS-tagging and dependency parsing, studying the influence of joint model and character features for parsing, Their model is extended from the arc-standard transition-based model, and can be regarded as an alternative to the arc-standard model of our work when pseudo intra-word dependencies are used. Similar work is done by Li and Zhou (2012). Our proposed arc-standard model is more concise while obtaining better performance than Hatori et al. (2012)'s work. With respect to word structures, real intra-word dependencies are often more complicated, while pseudo word structures cannot be used to correctly guide segmentation.

Zhao (2009), Hatori et al. (2012) and our work all study character-level dependency parsing. While Zhao (2009) focus on word internal structures using pseudo inter-word dependencies, Hatori et al. (2012) investigate a joint model using pseudo intra-word dependencies. We use manual dependencies for both inner- and inter-word structures, studying their influences on each other.

Zhang et al. (2013) was the first to perform Chinese syntactic parsing over characters. They extended word-level constituent trees by annotated word structures, and proposed a transition-based approach to parse intra-word structures and word-level constituent structures jointly. For Hebrew, Tsarfaty and Goldberg (2008) investigated joint segmentation and parsing over characters using a graph-based method. Our work is similar in exploiting character-level syntax. We study the dependency grammar, another popular syntactic representation, and propose two novel transition systems for character-level dependency parsing.

Nivre (2008) gave a systematic description of the arc-standard and arc-eager algorithms, currently two popular transition-based parsing methods for word-level dependency parsing. We extend both algorithms to character-level joint word segmentation, POS-tagging and dependency parsing. To our knowledge, we are the first to apply the arc-eager system to joint models and achieve comparative performances to the arc-standard model.

6 Conclusions

We studied the character-level Chinese dependency parsing, by making novel extensions to two commonly-used transition-based dependency parsing algorithms for word-based dependency parsing. With both pseudo and annotated word structures, our character-level models obtained better accuracies than previous work on segmentation, POS-tagging and word-level dependency parsing. We further analyzed some important factors for intra-word dependencies, and found that two proposed character-level parsing models are complementary in parsing intra-word dependencies. We make the source code publicly available at `http://sourceforge.net/projects/zpar/, version 0.7`.

Acknowledgments

We thank the anonymous reviewers for their constructive comments, and gratefully acknowledge the support of the National Basic Research Program (973 Program) of China via Grant 2014CB340503, the National Natural Science Foundation of China (NSFC) via Grant 61133012 and 61370164, the Singapore Ministry of Education (MOE) AcRF Tier 2 grant T2MOE201301 and SRG ISTD 2012 038 from Singapore University of Technology and Design.

References

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the EMNLP-CONLL*, pages 1455–1465, Jeju Island, Korea, July.

Bernd Bohnet. 2010. Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd COLING*, number August, pages 89–97.

Xavier Carreras, Mihai Surdeanu, and Lluís Màrquez. 2006. Projective dependency parsing with perceptron. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 181–185, New York City, June.

Pi-Chuan Chang, Michel Galley, and Chris Manning. 2008. Optimizing chinese word segmentation for machine translation performance. In *ACL Workshop on Statistical Machine Translation*.

Pi-Chuan Chang, Huihsin Tseng, Dan Jurafsky, , and Christopher D. Manning. 2009. Discriminative reordering with chinese grammatical relations features. In *Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation*.

Jinho D. Choi and Andrew McCallum. 2013. Transition-based dependency parsing with selectional branching. In *Proceedings of ACL*, pages 1052–1062, August.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 111–118, Barcelona, Spain, July.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 7th EMNLP*.

Xiangyu Duan, Jun Zhao, and Bo Xu. 2007. Probabilistic models for action-based chinese dependency parsing. In *Proceedings of ECML/ECPPKDD*, volume 4701 of *Lecture Notes in Computer Science*, pages 559–566.

Thomas Emerson. 2005. The second international chinese word segmentation bakeoff. In *Proceedings of the Second SIGHAN Workshop on Chinese Language Processing*, pages 123–133.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2012. Incremental joint approach to word segmentation, pos tagging, and dependency parsing in chinese. In *Proceedings of the 50th ACL*, pages 1045–1053, Jeju Island, Korea, July.

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th ACL*, pages 1077–1086, Uppsala, Sweden, July.

Liang Huang, Wenbin Jiang, and Qun Liu. 2009. Bilingually-constrained (monolingual) shift-reduce parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1222–1231. Association for Computational Linguistics.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the ACL*, pages 1–11.

Zhongguo Li and Guodong Zhou. 2012. Unified dependency parsing of chinese morphological and syntactic structures. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1445–1454, Jeju Island, Korea, July.

Zhenghua Li, Ting Liu, and Wanxiang Che. 2012. Exploiting multiple treebanks for parsing with quasi-synchronous grammars. In *Proceedings of the 50th ACL*, pages 675–684, Jeju Island, Korea, July.

Zhongguo Li. 2011. Parsing the internal structure of words: A new paradigm for chinese word segmentation. In *Proceedings of the 49th ACL*, pages 1405–1414, Portland, Oregon, USA, June.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL*, number June, pages 91–98, Morristown, NJ, USA.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of ACL*.

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

Reut Tsarfaty and Yoav Goldberg. 2008. Word-based or morpheme-based? annotation strategies for modern hebrew clitics. In *LREC*. European Language Resources Association.

Yiou Wang, Jun'ichi Kazama, Yoshimasa Tsuruoka, Wenliang Chen, Yujie Zhang, and Kentaro Torisawa. 2011. Improving chinese word segmentation and pos tagging with semi-supervised methods using large auto-analyzed data. In *Proceedings of 5th IJCNLP*, pages 309–317, Chiang Mai, Thailand, November.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of EMNLP*, pages 562–571, Honolulu, Hawaii, October.

Yue Zhang and Stephen Clark. 2010. A fast decoder for joint word segmentation and POS-tagging using a single discriminative model. In *Proceedings of the EMNLP*, pages 843–852, Cambridge, MA, October.

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

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th ACL*, pages 188–193, Portland, Oregon, USA, June.

Ruiqiang Zhang, Keiji Yasuda, and Eiichiro Sumita. 2008. Chinese word segmentation and statistical machine translation. *IEEE Transactions on Signal Processing*, 5(2).

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2013. Chinese parsing exploiting characters. In *Proceedings of the 51st ACL*, pages 125–134, Sofia, Bulgaria, August.

Hai Zhao. 2009. Character-level dependencies in chinese: Usefulness and learning. In *Proceedings of the EACL*, pages 879–887, Athens, Greece, March.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st ACL*, pages 434–443, Sofia, Bulgaria, August.

Unsupervised Dependency Parsing with Transferring Distribution via Parallel Guidance and Entropy Regularization

Xuezhe Ma
Department of Linguistics
University of Washington
Seattle, WA 98195, USA
xzma@uw.edu

Fei Xia
Department of Linguistics
University of Washington
Seattle, WA 98195, USA
fxia@uw.edu

Abstract

We present a novel approach for inducing unsupervised dependency parsers for languages that have no labeled training data, but have translated text in a resource-rich language. We train probabilistic parsing models for resource-poor languages by transferring cross-lingual knowledge from resource-rich language with entropy regularization. Our method can be used as a purely monolingual dependency parser, requiring no human translations for the test data, thus making it applicable to a wide range of resource-poor languages. We perform experiments on three Data sets — Version 1.0 and version 2.0 of Google Universal Dependency Treebanks and Treebanks from CoNLL shared-tasks, across ten languages. We obtain state-of-the art performance of all the three data sets when compared with previously studied unsupervised and projected parsing systems.

1 Introduction

In recent years, dependency parsing has gained universal interest due to its usefulness in a wide range of applications such as synonym generation (Shinyama et al., 2002), relation extraction (Nguyen et al., 2009) and machine translation (Katz-Brown et al., 2011; Xie et al., 2011). Several supervised dependency parsing algorithms (Nivre and Scholz, 2004; McDonald et al., 2005a; McDonald et al., 2005b; McDonald and Pereira, 2006; Carreras, 2007; Koo and Collins, 2010; Ma and Zhao, 2012; Zhang et al., 2013) have been proposed and achieved high parsing accuracies on several treebanks, due in large part to the availability of dependency treebanks in a number of languages (McDonald et al., 2013). However, the manually annotated treebanks that these parsers rely on are highly expensive to create, in particular when we want to build treebanks for resource-poor languages. This led to a vast amount of research on unsupervised grammar induction (Carroll and Charniak, 1992; Klein and Manning, 2004; Smith and Eisner, 2005; Cohen and Smith, 2009; Spitkovsky et al., 2010; Blunsom and Cohn, 2010; Mareček and Straka, 2013; Spitkovsky et al., 2013), which appears to be a natural solution to this problem, as unsupervised methods require only unannotated text for training parsers. Unfortunately, the unsupervised grammar induction systems' parsing accuracies often significantly fall behind those of supervised systems (McDonald et al., 2011). Furthermore, from a practical standpoint, it is rarely the case that we are completely devoid of resources for most languages.

In this paper, we consider a practically motivated scenario, in which we want to build statistical parsers for resource-poor target languages, using existing resources from a resource-rich source language (like English).[1] We assume that there are absolutely no labeled training data for the target language, but we have access to parallel data with a resource-rich language and a sufficient amount of labeled training data to build an accurate parser for the resource-rich language. This scenario appears similar to the setting in bilingual text parsing. However, most bilingual text parsing approaches require bilingual treebanks — treebanks that have manually annotated tree structures on both sides of source and target languages (Smith and Smith, 2004; Burkett and Klein, 2008), or have tree structures on the source side and translated sentences in the target languages (Huang et

[1] For the sake of simplicity, we refer to the resource-poor language as the "target language", and resource-rich language as the "source language". In addition, in this study we use English as the source resource-rich language, but our methodology can be applied to any resource-rich languages.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1337–1348,
Baltimore, Maryland, USA, June 23-25 2014.

al., 2009; Chen et al., 2010). Obviously, bilingual treebanks are much more difficult to acquire than the resources required in our scenario, since the labeled training data and the parallel text in our case are completely separated. What is more important is that most studies on bilingual text parsing assumed that the parser is applied only on bilingual text. But our goal is to develop a parser that can be used in completely monolingual setting for each target language of interest.

This scenario is applicable to a large set of languages and many research studies (Hwa et al., 2005) have been made on it. Ganchev et al. (2009) presented a parser projection approach via parallel text using the posterior regularization framework (Graca et al., 2007). McDonald et al. (2011) proposed two parser transfer approaches between two different languages — one is directly transferred parser from delexicalized parsers, and the other parser is transferred using constraint driven learning algorithm where constraints are drawn from parallel corpora. In that work, they demonstrate that even the directly transferred delexicalized parser produces significantly higher accuracies than unsupervised parsers. Cohen et al. (2011) proposed an approach for unsupervised dependency parsing with non-parallel multilingual guidance from one or more helper languages, in which parallel data is not used.

In this work, we propose a learning framework for transferring dependency grammars from a resource-rich language to resource-poor languages via parallel text. We train probabilistic parsing models for resource-poor languages by maximizing a combination of likelihood on parallel data and confidence on unlabeled data. Our work is based on the learning framework used in Smith and Eisner (2007), which is originally designed for parser bootstrapping. We extend this learning framework so that it can be used to transfer cross-lingual knowledge between different languages.

Throughout this paper, English is used as the source language and we evaluate our approach on ten target languages — Danish (da), Dutch (nl), French (fr), German (de), Greek (el), Italian (it), Korean (ko), Portuguese (pt), Spanish (es) and Swedish (sv). Our approach achieves significant improvement over previous state-of-the-art unsupervised and projected parsing systems across all the ten languages, and considerably bridges the gap to fully supervised dependency parsing performance.

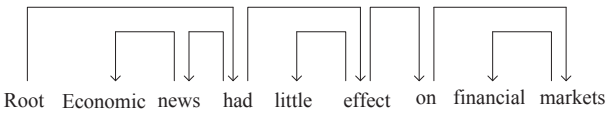


Figure 1: An example dependency tree.

2 Our Approach

Dependency trees represent syntactic relationships through labeled directed edges between heads and their dependents. For example, Figure 1 shows a dependency tree for the sentence, *Economic news had little effect on financial markets*, with the sentence's root-symbol as its root. The focus of this work is on building dependency parsers for target languages, assuming that an accurate English dependency parser and some parallel text between the two languages are available. Central to our approach is a maximizing likelihood learning framework, in which we use an English parser and parallel text to estimate the "transferring distribution" of the target language parsing model (See Section 2.2 for more details). Another advantage of the learning framework is that it combines both the likelihood on parallel data and confidence on unlabeled data, so that both parallel text and unlabeled data can be utilized in our approach.

2.1 Edge-Factored Parsing Model

In this paper, we will use the following notation: $\boldsymbol{x}$ represents a generic input sentence, and $\boldsymbol{y}$ represents a generic dependency tree. $\mathrm{T}(\boldsymbol{x})$ is used to denote the set of possible dependency trees for sentence $\boldsymbol{x}$. The probabilistic model for dependency parsing defines a family of conditional probability $p_\lambda(\boldsymbol{y}|\boldsymbol{x})$ over all $\boldsymbol{y}$ given sentence $\boldsymbol{x}$, with a log-linear form:

$$p_\lambda(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \exp\left\{\sum_j \lambda_j F_j(\boldsymbol{y}, \boldsymbol{x})\right\} \quad (1)$$

where F_j are feature functions, $\lambda = (\lambda_1, \lambda_2, \ldots)$ are parameters of the model, and $Z(\boldsymbol{x})$ is a normalization factor, which is commonly referred to as the *partition function*:

$$Z(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in \mathrm{T}(\boldsymbol{x})} \exp\left\{\sum_j \lambda_j F_j(\boldsymbol{y}, \boldsymbol{x})\right\} \quad (2)$$

A common strategy to make this parsing model efficiently computable is to *factor* dependency trees into sets of edges:

$$F_j(\boldsymbol{y}, \boldsymbol{x}) = \sum_{e \in y} f_j(e, \boldsymbol{x}). \quad (3)$$

That is, dependency tree y is treated as a set of edges e and each feature function $F_j(\boldsymbol{y}, \boldsymbol{x})$ is equal to the sum of all the features $f_j(e, \boldsymbol{x})$.

We denote the *weight function* of each edge e as follows:

$$w(e, \boldsymbol{x}) = \exp\left\{ \sum_j \lambda_j f_j(e, \boldsymbol{x}) \right\} \quad (4)$$

and the conditional probability $p_\lambda(\boldsymbol{y}|\boldsymbol{x})$ has the following form:

$$p_\lambda(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \prod_{e \in y} w(e, \boldsymbol{x}) \quad (5)$$

2.2 Model Training

One of the most common model training methods for supervised dependency parser is Maximum conditional likelihood estimation. For a supervised dependency parser with a set of training data $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$, the logarithm of the likelihood (a.k.a. the log-likelihood) is given by:

$$L(\lambda) = \sum_i \log p_\lambda(\boldsymbol{y}_i|\boldsymbol{x}_i) \quad (6)$$

Maximum likelihood training chooses parameters such that the log-likelihood $L(\lambda)$ is maximized.

However, in our scenario we have no labeled training data for target languages but we have some parallel and unlabeled data plus an English dependency parser. For the purpose of transferring cross-lingual information from the English parser via parallel text, we explore the model training method proposed by Smith and Eisner (2007), which presented a generalization of K function (Abney, 2004), and related it to another semi-supervised learning technique, entropy regularization (Jiao et al., 2006; Mann and McCallum, 2007). The objective K function to be minimized is actually the *expected* negative log-likelihood:

$$\begin{aligned} K &= -\sum_i \sum_{\boldsymbol{y}_i} \tilde{p}(\boldsymbol{y}_i|\boldsymbol{x}_i) \log p_\lambda(\boldsymbol{y}_i|\boldsymbol{x}_i) \\ &= \sum_i D(\tilde{p}_i || p_{\lambda,i}) + H(\tilde{p}_i) \end{aligned} \quad (7)$$

where $\tilde{p}_i(\cdot) \stackrel{def}{=} \tilde{p}(\cdot|\boldsymbol{x}_i)$ and $p_{\lambda,i}(\cdot) \stackrel{def}{=} p_\lambda(\cdot|\boldsymbol{x}_i)$. $\tilde{p}(\boldsymbol{y}|\boldsymbol{x})$ is the "transferring distribution" that reflects our uncertainty about the true labels, and we are trying to learn a parametric model $p_\lambda(\boldsymbol{y}|\boldsymbol{x})$ by minimizing the K function.

In our scenario, we have a set of aligned parallel data $P = \{\boldsymbol{x}_i^s, \boldsymbol{x}_i^t, a_i\}$ where a_i is the word alignment for the pair of source-target sentences $(\boldsymbol{x}_i^s, \boldsymbol{x}_i^t)$, and a set of unlabeled sentences of the target language $U = \{\boldsymbol{x}_i^t\}$. We also have a trained English parsing model $p_{\lambda_E}(\boldsymbol{y}|\boldsymbol{x})$. Then the K in equation (7) can be divided into two cases, according to whether $\boldsymbol{x}_i$ belongs to parallel data set P or unlabeled data set U. For the unlabeled examples $\{\boldsymbol{x}_i \in U\}$, some previous studies (e.g., (Abney, 2004)) simply use a uniform distribution over labels (e.g., parses), to reflect that the label is unknown. We follow the method in Smith and Eisner (2007) and take the transferring distribution $\tilde{p}_i$ to be the *actual* current belief $p_{\lambda,i}$. The total contribution of the *unsupervised* examples to K then simplifies to $K_U = \sum_{\boldsymbol{x}_i \in U} H(p_{\lambda,i})$, which may be regarded as the entropy item used to constrain the model's uncertainty H to be low, as presented in the work on entropy regularization (Jiao et al., 2006; Mann and McCallum, 2007).

But how can we define the transferring distribution for the parallel examples $\{\boldsymbol{x}_i^t \in P\}$? We define the transferring distribution by defining the *transferring weight* utilizing the English parsing model $p_{\lambda_E}(\boldsymbol{y}|\boldsymbol{x})$ via parallel data with word alignments:

$$\tilde{w}(e^t, \boldsymbol{x}_i^t) = \begin{cases} w_E(e^s, \boldsymbol{x}_i^s), & \text{if } e^t \stackrel{align}{\longrightarrow} e^s \\ w_E(e_{delex}^t, \boldsymbol{x}_i^s), & \text{otherwise} \end{cases} \quad (8)$$

where $w_E(\cdot,\cdot)$ is the weight function of the English parsing model $p_{\lambda_E}(\boldsymbol{y}|\boldsymbol{x})$, and e_{delex}^t is the delexicalized form[2] of the edge e^t. From the definition of the transferring weight, we can see that, if an edge e^t of the target language sentence $\boldsymbol{x}_i^t$ is aligned to an edge e^s of the English sentence $\boldsymbol{x}_i^s$, we transfer the weight of edge e^t to the corresponding weight of edge e^s in the English parsing model $p_{\lambda_E}(\boldsymbol{y}|\boldsymbol{x})$. If the edge e^t is not aligned to any edges of the English sentence $\boldsymbol{x}_i^s$, we reduce the edge e^t to the delexicalized form and calculate the transferring weight in the English parsing model. There are two advan-

[2]The delexicalized form of an edge is an edge for which only delexicalized features are considered.

tages for this definition of the transferring weight. First, by transferring the weight function to the corresponding weight in the well-developed English parsing model, we can project syntactic information across language boundaries. Second, McDonald et al. (2011) demonstrates that parsers with only delexicalized features produce considerably high parsing performance. By reducing unaligned edges to their delexicalized forms, we can still use those delexicalized features, such as part-of-speech tags, for those unaligned edges, and can address problem that automatically generated word alignments include errors.

From the definition of transferring weight in equation (8), the transferring distribution can be defined in the following way:

$$\tilde{p}(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{\tilde{Z}(\boldsymbol{x})} \prod_{e \in y} \tilde{w}(e, \boldsymbol{x}) \tag{9}$$

where

$$\tilde{Z}(\boldsymbol{x}) = \sum_{y} \prod_{e \in y} \tilde{w}(e, \boldsymbol{x}) \tag{10}$$

Due to the normalizing factor $\tilde{Z}(\boldsymbol{x})$, the transferring distribution is a valid one.

We introduce a multiplier γ as a trade-off between the two contributions (parallel and unsupervised) of the objective function K, and the final objective function $K^{'}$ has the following form:

$$\begin{aligned} K^{'} &= -\sum_{x_i \in P} \sum_{y_i} \tilde{p}(\boldsymbol{y}_i|\boldsymbol{x}_i) \log p_\lambda(\boldsymbol{y}_i|\boldsymbol{x}_i) \\ &\quad + \gamma \sum_{x_i \in U} H(p_{\lambda,i}) \\ &= K_P + \gamma K_U \end{aligned} \tag{11}$$

K_P and K_U are the contributions of the parallel and unsupervised data, respectively. One may regard γ as a Lagrange multiplier that is used to constrain the parser's uncertainty H to be low, as presented in several studies on entropy regularization (Brand, 1998; Grandvalet and Bengio, 2004; Jiao et al., 2006).

2.3 Algorithms and Complexity for Model Training

To train our parsing model, we need to find out the parameters λ that minimize the objective function $K^{'}$ in equation (11). This optimization problem is typically solved using quasi-Newton numerical methods such as L-BFGS (Nash and Nocedal, 1991), which requires efficient calculation of the objective function and the gradient of the objective function.

The first item (K_P) of the $K^{'}$ function in equation (11) can be rewritten in the following form:

$$\begin{aligned} K_P &= -\sum_{\boldsymbol{x}_i \in P} \Big[\sum_{y_i} \tilde{p}(\boldsymbol{y}_i|\boldsymbol{x}_i) \sum_{e \in \boldsymbol{y}_i} \log w(e, \boldsymbol{x}_i) \\ &\quad - \log Z(\boldsymbol{x}_i) \Big] \end{aligned} \tag{12}$$

and according to equation (1) and (3) the gradient of K_P can be written as:

$$\begin{aligned} \frac{\partial K_P}{\partial \lambda_j} &= \sum_{x_i \in P} \frac{\partial \tilde{p}(\boldsymbol{y}_i|\boldsymbol{x}_i) \log p_\lambda(\boldsymbol{y}_i|\boldsymbol{x}_i)}{\partial \lambda_j} \\ &= \sum_{x_i \in P} \Bigg[\sum_{y_i} \tilde{p}(\boldsymbol{y}_i|\boldsymbol{x}_i) \sum_{e \in y_i} f_j(e, \boldsymbol{x}_i) \\ &\quad - \sum_{y_i} p_\lambda(\boldsymbol{y}_i|\boldsymbol{x}_i) \sum_{e \in y_i} f_j(e, \boldsymbol{x}_i) \Bigg] \end{aligned} \tag{13}$$

According to equation (9), $\tilde{p}(\boldsymbol{y}|\boldsymbol{x})$ can also be factored into the multiplication of the weight of each edge, so both K_P and its gradient can be calculated by running the $O(n^3)$ inside-outside algorithm (Baker, 1979; Paskin, 2001) for projective parsing. For non-projective parsing, the analogy to the inside algorithm is the $O(n^3)$ matrix-tree algorithm based on Kirchhoff's Matrix-Tree Theorem, which is dominated asymptotically by a matrix determinant (Koo et al., 2007; Smith and Smith, 2007). The gradient of a determinant may be computed by matrix inversion, so evaluating the gradient again has the same $O(n^3)$ complexity as evaluating the function.

The second item (K_U) of the $K^{'}$ function in equation (11) is the Shannon entropy of the posterior distribution over parsing trees, and can be written into the following form:

$$\begin{aligned} K_U &= -\sum_{\boldsymbol{x}_i \in U} \Big[\sum_{y_i} p_\lambda(\boldsymbol{y}_i|\boldsymbol{x}_i) \sum_{e \in \boldsymbol{y}_i} \log w(e, \boldsymbol{x}_i) \\ &\quad - \log Z(\boldsymbol{x}_i) \Big] \end{aligned} \tag{14}$$

and the gradient of K_U is in the following:

$$\begin{aligned} \frac{\partial K_U}{\partial \lambda_j} &= \sum_{x_i \in U} \frac{\partial p_\lambda(\boldsymbol{y}_i|\boldsymbol{x}_i) \log p_\lambda(\boldsymbol{y}_i|\boldsymbol{x}_i)}{\partial \lambda_j} \\ &= -\sum_{y_i} p_\lambda(\boldsymbol{y}_i|\boldsymbol{x}_i) \log p_\lambda(\boldsymbol{y}_i|\boldsymbol{x}_i) F_j(\boldsymbol{y}_i, \boldsymbol{x}_i) \\ &\quad + \Big(\sum_{y_i} p_\lambda(\boldsymbol{y}_i|\boldsymbol{x}_i) \log p_\lambda(\boldsymbol{y}_i|\boldsymbol{x}_i) \Big) \\ &\quad \cdot \Big(\sum_{y_i} p_\lambda(\boldsymbol{y}_i|\boldsymbol{x}_i) F_j(\boldsymbol{y}_i, \boldsymbol{x}_i) \Big) \end{aligned} \tag{15}$$

	#sents/#tokens		
	training	dev	test
	Version 1.0		
de	2,200/30,460	800/12,215	1,000/16,339
es	3,345/94,232	370/10,191	300/8,295
fr	3,312/74,979	366/8,071	300/6,950
ko	5,308/62,378	588/6,545	298/2,917
sv	4,447/66,631	493/9,312	1,219/20,376
	Version 2.0		
de	14,118/26,4906	800/12,215	1,000/16,339
es	14,138/37,5180	1,569/40,950	300/8,295
fr	14,511/35,1233	1,611/38,328	300/6,950
it	6,389/14,9145	400/9,541	400/9,187
ko	5437/60,621	603/6,438	299/2,631
pt	9,600/23,9012	1,200/29,873	1,198/29,438
sv	4,447/66,631	493/9,312	1,219/20,376

Table 1: Data statistics of two versions of Google Universal Treebanks for the target languages.

Similar with the calculation of K_P, K_U can also be computed by running the inside-outside algorithm (Baker, 1979; Paskin, 2001) for projective parsing. For the gradient of K_U, both the two multipliers of the second item in equation (15) can be computed using the same inside-outside algorithm. For the first item in equation (15), an $O(n^3)$ dynamic programming algorithm that is closely related to the forward-backward algorithm (Mann and McCallum, 2007) for the entropy regularized CRF (Jiao et al., 2006) can be used for projective parsing. For non-projective parsing, however, the runtime rises to $O(n^4)$. In this paper, we focus on projective parsing.

2.4 Summary of Our Approach

To summarize the description in the previous sections, our approach is performed in the following steps:

1. Train an English parsing model $p_{\lambda_E}(\boldsymbol{y}|\boldsymbol{x})$, which is used to estimate the transferring distribution $\tilde{p}(\boldsymbol{y}|\boldsymbol{x})$.

2. Prepare parallel text by running word alignment method to obtain word alignments,[3] and prepare the unlabeled data.

3. Train a parsing model for the target language by minimizing the objective K' function which is the combination of expected negative log-likelihood on parallel and unlabeled data.

[3]The word alignment methods do not require additional resources besides parallel text.

	# sents					
	500	1000	2000	5000	10000	20000
da	12,568	25,225	49,889	126,623	254,565	509,480
de	13,548	26,663	53,170	133,596	265,589	527,407
el	14,198	28,302	56,744	143,753	286,126	572,777
es	15,147	29,214	57,526	144,621	290,517	579,164
fr	15,046	29,982	60,569	153,874	306,332	609,541
it	15,151	29,786	57,696	145,717	288,337	573,557
ko	3,814	7,679	15,337	38,535	77,388	155,051
nl	13,234	26,777	54,570	137,277	274,692	551,463
pt	14,346	28,109	55,998	143,221	285,590	571,109
sv	12,242	24,897	50,047	123,069	246,619	490,086

Table 2: The number of tokens in parallel data used in our experiments. For all these corpora, the other language is English.

3 Data and Tools

In this section, we illustrate the data sets used in our experiments and the tools for data preparation.

3.1 Choosing Target Languages

Our experiments rely on two kinds of data sets: (i) Monolingual Treebanks with consistent annotation schema — English treebank is used to train the English parsing model, and the Treebanks for target languages are used to evaluate the parsing performance of our approach. (ii) Large amounts of parallel text with English on one side. We select target languages based on the availability of these resources. The monolingual treebanks in our experiments are from the Google Universal Dependency Treebanks (McDonald et al., 2013), for the reason that the treebanks of different languages in Google Universal Dependency Treebanks have consistent syntactic representations.

The parallel data come from the Europarl corpus version 7 (Koehn, 2005) and Kaist Corpus[4]. Taking the intersection of languages in the two kinds of resources yields the following seven languages: French, German, Italian, Korean, Portuguese, Spanish and Swedish.

The treebanks from CoNLL shared-tasks on dependency parsing (Buchholz and Marsi, 2006; Nivre et al., 2007) appear to be another reasonable choice. However, previous studies (McDonald et al., 2011; McDonald et al., 2013) have demonstrated that a homogeneous representation is critical for multilingual language technologies that require consistent cross-lingual analysis for downstream components, and the heterogenous representations used in CoNLL shared-tasks treebanks weaken any conclusion that can be drawn.

[4]`http://semanticweb.kaist.ac.kr/home/index.php/Corpus10`

	DTP	DTP†	PTP†	-U	+U	OR
de	58.50	58.46	69.21	73.72	74.01	78.64
es	68.07	68.72	72.57	75.32	75.60	82.56
fr	70.14	71.13	74.60	76.65	76.93	83.69
ko	42.37	43.57	53.72	59.72	59.94	89.85
sv	70.56	70.59	75.87	78.91	79.27	85.59
Ave	61.93	62.49	69.19	72.86	73.15	84.67

Table 3: UAS for two versions of our approach, together with baseline and oracle systems on Google Universal Treebanks version 1.0. "Ave" is the macro-average across the five languages.

For comparison with previous studies, nevertheless, we also run experiments on CoNLL treebanks (see Section 4.4 for more details). We evaluate our approach on three target languages from CoNLL shared task treebanks, which do not appear in Google Universal Treebanks. The three languages are Danish, Dutch and Greek. So totally we have ten target languages. The parallel data for these three languages are also from the Europarl corpus version 7.

3.2 Word Alignments

In our approach, word alignments for the parallel text are required. We perform word alignments with the open source GIZA++ toolkit[5]. The parallel corpus was preprocessed in standard ways, selecting sentences with the length in the range from 3 to 100. Then we run GIZA++ with the default setting to generate word alignments in both directions. We then make the intersection of the word alignments of two directions to generate one-to-one alignments.

3.3 Part-of-Speech Tagging

Several features in our parsing model involve part-of-speech (POS) tags of the input sentences. The set of POS tags needs to be consistent across languages and treebanks. For this reason we use the universal POS tag set of Petrov et al. (2011). This set consists of the following 12 coarse-grained tags: NOUN (nouns), VERB (verbs), ADJ (adjectives), ADV (adverbs), PRON (pronouns), DET (determiners), ADP (prepositions or postpositions), NUM (numerals), CONJ (conjunctions), PRT (particles), PUNC (punctuation marks) and X (a catch-all for other categories such as abbreviations or foreign words).

POS tags are not available for parallel data in the Europarl and Kaist corpus, so we need to provide the POS tags for these data. In our experiments, we train a Stanford POS Tagger (Toutanova et al., 2003) for each language. The labeled training data for each POS tagger are extracted from the training portion of each Treebanks. The average tagging accuracy is around 95%.

Undoubtedly, we are primarily interested in applying our approach to build statistical parsers for resource-poor target languages without any knowledge. For the purpose of evaluation of our approach and comparison with previous work, we need to exploit the gold POS tags to train the POS taggers. As part-of-speech tags are also a form of syntactic analysis, this assumption weakens the applicability of our approach. Fortunately, some recently proposed POS taggers, such as the POS tagger of Das and Petrov (2011), rely only on labeled training data for English and the same kind of parallel text in our approach. In practice we can use this kind of POS taggers to predict POS tags, whose tagging accuracy is around 85%.

[5] https://code.google.com/p/giza-pp/

	DTP†	PTP†	-U	+U	OR
de	58.56	69.77	73.92	74.30	81.65
es	68.72	73.22	75.21	75.53	83.92
fr	71.13	74.75	76.14	76.53	83.51
it	70.74	76.08	77.55	77.74	85.47
ko	38.55	43.34	59.71	59.89	90.42
pt	69.82	74.59	76.30	76.65	85.67
sv	70.59	75.87	78.91	79.27	85.59
Ave	64.02	69.66	73.96	74.27	85.18

Table 4: UAS for two versions of our approach, together with baseline and oracle systems on Google Universal Treebanks version 2.0. "Ave" is the macro-average across the seven languages.

4 Experiments

In this section, we will describe the details of our experiments and compare our results with previous methods.

4.1 Data Sets

As presented in Section 3.1, we evaluate our parsing approach on both version 1.0 and version 2.0 of Google Univereal Treebanks for seven languages[6]. We use the standard splits of the treebank for each language as specified in the release of the data[7]. Table 1 presents the statistics of the two versions of Google Universal Treebanks. We strip all

[6] Japanese and Indonesia are excluded as no practicable parallel data are available.

[7] https://code.google.com/p/uni-dep-tb/

	Google Universal Treebanks V1.0														
	de			**es**			**fr**			**ko**			**sv**		
# sents	PTP†	-U	+U	PTP†	-U	+U	PTP†	-U	+U	PTP†	-U	+U	PTP†	-U	+U
500	63.23	70.79	70.93	70.09	72.32	72.64	72.24	74.64	74.90	47.71	56.87	57.22	71.70	75.88	76.13
1000	65.61	71.71	71.86	70.90	73.44	73.67	72.95	75.07	75.35	47.83	57.65	58.15	72.38	76.55	77.03
2000	66.52	72.33	72.48	72.01	73.57	73.81	73.69	75.88	76.22	48.37	58.19	58.44	73.65	77.86	78.12
5000	67.79	73.06	73.31	72.34	74.30	74.79	74.31	76.02	76.29	53.02	58.57	59.04	74.88	78.48	78.70
10000	68.44	73.59	73.92	72.48	74.86	75.26	74.43	76.14	76.34	53.61	59.17	59.55	75.34	78.78	79.08
20000	69.21	73.72	74.01	72.57	75.32	75.60	74.60	76.55	76.93	53.72	59.72	59.94	75.87	78.91	79.27

	Google Universal Treebanks V2.0														
	de			**es**			**fr**			**ko**			**it**		
# sents	PTP†	-U	+U	PTP†	-U	+U	PTP†	-U	+U	PTP†	-U	+U	PTP†	-U	+U
500	60.10	71.07	71.39	69.52	72.97	73.28	71.10	74.57	74.70	40.09	56.60	57.10	72.80	75.67	75.94
1000	61.76	72.15	72.39	70.78	73.48	73.79	72.14	75.13	75.43	40.44	57.55	57.93	73.55	76.43	76.67
2000	65.35	72.73	73.04	71.75	74.10	74.35	73.21	75.78	76.06	40.87	58.11	58.43	74.44	76.99	77.39
5000	67.86	73.32	73.62	72.43	74.55	74.83	74.14	75.83	76.02	40.90	58.48	58.96	75.07	77.10	77.34
10000	68.70	73.71	74.02	72.85	74.80	74.95	74.53	75.97	76.17	41.29	59.13	59.44	75.65	77.50	77.71
20000	69.77	73.92	74.30	73.22	75.21	75.53	74.75	76.14	76.53	43.34	59.71	59.89	76.08	77.55	77.74

	pt		
# sents	PTP†	-U	+U
500	71.34	74.41	74.68
1000	71.91	74.48	75.08
2000	72.93	75.10	75.32
5000	73.78	75.88	75.98
10000	74.40	75.99	76.15
20000	74.59	76.30	76.65

Table 5: Parsing results of our approach with different amount of parallel data on Google Universal Treebanks version 1.0 and 2.0. We omit the results of Swedish for treebanks version 2.0 since the data for Swedish from version 2.0 are exactly the same with those from version 1.0.

the dependency annotations off the training portion of each treebank, and use that as the unlabeled data for that target language. We train our parsing model with different numbers of parallel sentences to analyze the influence of the amount of parallel data on the parsing performance of our approach. The parallel data sets contain 500, 1000, 2000, 5000, 10000 and 20000 parallel sentences, respectively. We randomly extract parallel sentences from each corpora, and smaller data sets are subsets of larger ones. Table 2 shows the number of tokens in the parallel data used in the experiments.

4.2 System performance and comparison on Google Universal Treebanks

For the comparison of parsing performance, we run experiments on the following systems:

DTP: The direct transfer parser (DTP) proposed by McDonald et al. (2011), who train a delexicalized parser on English labeled training data with no lexical features, then apply this parser to parse target languages directly. It is based on the transition-based dependency parsing paradigm (Nivre, 2008). We directly cite the results reported in McDonald et al. (2013). In addition to their original results, we also report results by reimplementing the direct transfer parser based on the first-order projective dependency parsing model (McDonald et al., 2005a) (DTP†).

PTP The projected transfer parser (PTP) described in McDonald et al. (2011). The results of the projected transfer parser reimplemented by us is marked as "PTP†".

-U: Our approach training on only parallel data without unlabeled data for the target language. The parallel data set for each language contains 20,000 sentences.

+U: Our approach training on both parallel and unlabeled data. The parallel data sets are the ones contains 20,000 sentences.

OR: the supervised first-order projective dependency parsing model (McDonald et al., 2005a), trained on the original treebanks with maximum likelihood estimation (equation 6). One may regard this system as an oracle of transfer parsing.

Parsing accuracy is measured with unlabeled attachment score (UAS): the percentage of words with the correct head.

Table 3 and Table 4 shows the parsing results of our approach, together with the results of the baseline systems and the oracle, on version 1.0 and version 2.0 of Google Universal Treebanks, respectively. Our approaches significantly outperform all the baseline systems across all the seven target languages. For the results on Google Universal Treebanks version 1.0, the improvement on average over the projected transfer paper (PTP†) is 3.96%

and up to 6.22% for Korean and 4.80% for German. For the other three languages, the improvements are remarkable, too — 2.33% for French, 3.03% for Spanish and 3.40% for Swedish. By adding entropy regularization from unlabeled data, our full model achieves average improvement of 0.29% over the "-U" setting. Moreover, our approach considerably bridges the gap to fully supervised dependency parsers, whose average UAS is 84.67%. For the results on treebanks version 2.0, we can get similar observation and draw the same conclusion.

4.3 Effect of the Amount of Parallel Text

Table 5 illustrates the UAS of our approach trained on different amounts of parallel data, together with the results of the projected transfer parser re-implemented by us (PTP†). We run two versions of our approach for each of the parallel data sets, one with unlabeled data (+U) and the other without them (-U). From table 5 we can get three observations. First, even the parsers trained with only 500 parallel sentences achieve considerably high parsing accuracies (average 70.10% for version 1.0 and 71.59% for version 2.0). This demonstrates that our approach does not rely on a large amount of parallel data. Second, when gradually increasing the amount of parallel data, the parsing performance continues improving. Third, entropy regularization with unlabeled data makes modest improvement on parsing performance over the parsers without unlabeled data. This proves the effectiveness of the entropy regularization from unlabeled data.

4.4 Experiments on CoNLL Treebanks

To make a thorough empirical comparison with previous studies, we also evaluate our system without unlabeled data (-U) on treebanks from CoNLL shared task on dependency parsing (Buchholz and Marsi, 2006; Nivre et al., 2007). To facilitate comparison, we use the same eight Indo-European languages as target languages: Danish, Dutch, German, Greek, Italian, Portuguese, Spanish and Swedish, and same experimental setup as McDonald et al. (2011). We report both the results of the direct transfer and projected transfer parsers directly cited from McDonald et al. (2011) (DTP and PTP) and re-implemented by us (DTP†and PTP†).

Table 6 gives the results comparing the model without unlabeled data (-U) presented in this work to those five baseline systems and the oracle (OR). The results of unsupervised DMV model (Klein and Manning, 2004) are from Table 1 of McDonald et al. (2011). Our approach outperforms all these baseline systems and achieves state-of-the-art performance on all the eight languages.

	DMV	DTP	DTP†	PTP	PTP†	-U	OR
da	33.4	45.9	46.8	48.2	50.0	50.1	87.1
de	18.0	47.2	46.0	50.9	52.4	57.3	87.0
el	39.9	63.9	62.9	66.8	65.3	67.4	82.3
es	28.5	53.3	54.4	55.8	59.9	60.3	83.6
it	43.1	57.7	59.9	60.8	63.4	64.0	83.9
nl	38.5	60.8	60.7	67.8	66.5	68.2	78.2
pt	20.1	69.2	71.1	71.3	74.8	75.1	87.2
sv	44.0	58.3	60.3	61.3	62.8	66.7	88.0
Ave	33.2	57.0	57.8	60.4	61.9	63.6	84.7

Table 6: Parsing results on treebanks from CoNLL shared tasks for eight target languages. The results of unsupervised DMV model are from Table 1 of McDonald et al. (2011).

In order to compare with more previous methods, we also report parsing performance on sentences of length 10 or less after punctuation has been removed. Table 7 shows the results of our system and the results of baseline systems. "USR†" is the weakly supervised system of Naseem et al. (2010). "PGI" is the phylogenetic grammar induction model of Berg-Kirkpatrick and Klein (2010). Both the results of the two systems are cited from Table 4 of McDonald et al. (2011). We also include the results of the unsupervised dependency parsing model with non-parallel multilingual guidance (NMG) proposed by Cohen et al. (2011)[8], and "PR" which is the posterior regularization approach presented in Gillenwater et al. (2010). All the results are shown in Table 7.

From Table 7, we can see that among the eight target languages, our approach achieves best parsing performance on six languages — Danish, German, Greek, Italian, Portuguese and Swedish. It should be noted that the "NMG" system utilizes more than one helper languages. So it is not directly comparable to our work.

4.5 Extensions

In this section, we briefly outline a few extensions to our approach that we want to explore in future work.

[8]For each language, we use the best result of the four systems in Table 3 of Cohen et al. (2011)

	DTP	DTP†	PTP	PTP†	USR†	PGI	PR	NMG	-U
da	53.2	55.3	57.4	59.8	55.1	41.6	44.0	59.9	**60.1**
de	65.9	57.9	67.0	63.5	60.0	—	—	—	**67.5**
el	73.9	70.8	73.9	72.3	60.3	—	—	73.0	**74.3**
es	58.0	62.3	62.3	66.1	68.3	58.4	62.4	**76.7**	64.6
it	65.5	66.9	69.9	71.5	47.9	—	—	—	**73.6**
nl	67.6	66.0	**72.2**	72.1	44.0	45.1	37.9	50.7	70.5
pt	77.9	79.2	80.6	82.9	70.9	63.0	47.8	79.8	**83.3**
sv	70.4	70.2	71.3	70.4	52.6	58.3	42.2	74.0	**75.1**
Ave	66.6	66.1	69.4	69.8	57.4	—	—	—	**71.1**

Table 7: UAS on sentences of length 10 or less without punctuation from CoNLL shared task treebanks. "USR†" is the weakly supervised system of Naseem et al. (2010). "PGI" is the phylogenetic grammar induction model of Berg-Kirkpatrick and Klein (2010). Both the "USR†" and "PGI" systems are implemented and reported by McDonald et al. (2011). "NMG" is the unsupervised dependency parsing model with non-parallel multilingual guidance (Cohen et al., 2011). "PR" is the posterior regularization approach presented in Gillenwater et al. (2010). Some systems' results for certain target languages are not available as marked by —.

4.5.1 Non-Projective Parsing

As mentioned in section 2.3, the runtime to compute K_U and its gradient is $O(n^4)$. One reasonable speedup, as presented in Smith and Eisner (2007), is to replace Shannon entropy with Rényi entropy. The **Rényi entropy** is parameterized by α:

$$R_\alpha(p) = \frac{1}{1-\alpha} \log \Big(\sum_y p(y)^\alpha \Big) \qquad (16)$$

With Rényi entropy, the computation of K_U and its gradient is $O(n^3)$, even for non-projective case.

4.5.2 Higher-Order Models for Projective Parsing

Our learning framework can be extended to higher-order dependency parsing models. For example, if we want to make our model capable of utilizing more contextual information, we can extend our transferring weight to higher-order parts:

$$\tilde{w}(p^t, \boldsymbol{x}_i^t) = \begin{cases} w_E(p^s, \boldsymbol{x}_i^s), & \text{if } p^t \overset{align}{\longrightarrow} p^s \\ w_E(p^t_{delex}, \boldsymbol{x}_i^s), & \text{otherwise} \end{cases} \qquad (17)$$

where p is a small *part* of tree $\boldsymbol{y}$ that has limited interactions. For projective parsing, several algorithms (McDonald and Pereira, 2006; Carreras, 2007; Koo and Collins, 2010; Ma and Zhao, 2012) have been proposed to solve the model training problems (calculation of objective function and gradient) for different factorizations.

4.5.3 IGT Data

One possible direction to improve our approach is to replace parallel text with Interlinear Glossed Text (IGT) (Lewis and Xia, 2010), which is a semi-structured data type encoding more syntactic information than parallel data. By using IGT Data, not only can we obtain more accurate word alignments, but also extract useful cross-lingual information for the resource-poor language.

5 Conclusion

In this paper, we propose an unsupervised projective dependency parsing approach for resource-poor languages, using existing resources from a resource-rich source language. By presenting a model training framework, our approach can utilize parallel text to estimate transferring distribution with the help of a well-developed resource-rich language dependency parser, and use unlabeled data as entropy regularization. The experimental results on three data sets across ten target languages show that our approach achieves significant improvement over previous studies.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. BCS-0748919. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Steven Abney. 2004. Understanding the Yarowsky algorithm. *Computational Linguistics*, 30:2004.

James K. Baker. 1979. Trainable grammars for speech recognition. In *Proceedings of 97th meeting of the Acoustical Society of America*, pages 547–550.

Taylor Berg-Kirkpatrick and Dan Klein. 2010. Phylogenetic grammar induction. In *Proceedings of ACL-2010*, pages 1288–1297, Uppsala, Sweden, July.

Phil Blunsom and Trevor Cohn. 2010. Unsupervised induction of tree substitution grammars for dependency parsing. In *Proceedings of EMNLP-2010*, pages 1204–1213, Cambridge, MA, October.

Matthew Brand. 1998. Structure learning in conditional probability models via an entropic prior and parameter extinction. *Neural Computation*, 11(5):1155–1182.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceeding of CoNLL-2006*, pages 149–164, New York, NY.

David Burkett and Dan Klein. 2008. Two languages are better than one (for syntactic parsing). In *Proceedings of EMNLP-2008*, pages 877–886, Honolulu, Hawaii, October.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CONLL*, pages 957–961.

Glenn Carroll and Eugene Charniak. 1992. Two experiments on learning probabilistic dependency grammars from corpora. In *Proceedings of Working Notes of the Workshop Statistically-Based NLP Techniques*.

Wenliang Chen, Jun'ichi Kazama, and Kentaro Torisawa. 2010. Bitext dependency parsing with bilingual subtree constraints. In *Proceedings of ACL-2010*, pages 21–29, Uppsala, Sweden, July.

Shay Cohen and Noah A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of NAACL/HLT-2009*, pages 74–82, Boulder, Colorado, June.

Shay B. Cohen, Dipanjan Das, and Noah A. Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *Proceedings of EMNLP-2011*, pages 50–61, Edinburgh, Scotland, UK., July.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of ACL/HLT-2011*, pages 600–609, Portland, Oregon, USA, June.

Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proceedings of ACL/AFNLP-2009*, pages 369–377, Suntec, Singapore, August.

Jennifer Gillenwater, Kuzman Ganchev, João Graça, Fernando Pereira, and Ben Taskar. 2010. Sparsity in dependency grammar induction. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 194–199, Uppsala, Sweden, July.

Joao V. Graca, Lf Inesc-id, Kuzman Ganchev, and Ben Taskar. 2007. Expectation maximization and posterior constraints. In *Advances in NIPS*, pages 569–576.

Yves Grandvalet and Yoshua Bengio. 2004. Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems*.

Liang Huang, Wenbin Jiang, and Qun Liu. 2009. Bilingually-constrained (monolingual) shift-reduce parsing. In *Proceedings of EMNLP-2009*, pages 1222–1231, Singapore, August.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*, 11:11–311.

Feng Jiao, Shaojun Wang, Chi-Hoon Lee, Russell Greiner, and Dale Schuurmans. 2006. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proceedings of COLING/ACL-2006*, pages 209–216, Sydney, Australia, July.

Jason Katz-Brown, Slav Petrov, Ryan McDonald, Franz Och, David Talbot, Hiroshi Ichikawa, Masakazu Seno, and Hideto Kazawa. 2011. Training a parser for machine translation reordering. In *Proceedings of EMNLP-2011*, pages 183–192, Edinburgh, Scotland, UK., July.

Dan Klein and Christopher Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of ACL-2004*, pages 478–485, Barcelona, Spain, July.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand. AAMT, AAMT.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of ACL-2010*, pages 1–11, Uppsala, Sweden, July.

Terry Koo, Amir Globerson, Xavier Carreras, and Michael Collins. 2007. Structured predicition models via the matrix-tree theorem. In *Proceedings of EMNLP-CONLL 2007*, pages 141–150, Prague, Czech, June.

William D. Lewis and Fei Xia. 2010. Developing odin: A multilingual repository of annotated language data for hundreds of the world's languages. *LLC*, 25(3):303–319.

Xuezhe Ma and Hai Zhao. 2012. Fourth-order dependency parsing. In *Proceedings of COLING 2012: Posters*, pages 785–796, Mumbai, India, December.

Gideon S. Mann and Andrew McCallum. 2007. Efficient computation of entropy gradient for semi-supervised conditional random fields. In *Proceedings of NAACL/HLT-2007*, pages 109–112, Stroudsburg, PA, USA.

David Mareček and Milan Straka. 2013. Stop-probability estimates computed on a large corpus improve unsupervised dependency parsing. In *Proceedings of ACL-2013*, pages 281–290, Sofia, Bulgaria, August.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL-2006*, pages 81–88, Trento, Italy, April.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of ACL-2005*, pages 91–98, Ann Arbor, Michigan, USA, June 25-30.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT/EMNLP-2005*, pages 523–530, Vancouver, Canada, October.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of EMNLP-2011*, pages 62–72, Edinburgh, Scotland, UK., July.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of ACL-2013*, pages 92–97, Sofia, Bulgaria, August.

Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proceedings of EMNLP-2010*, pages 1234–1244, Cambridge, MA, October.

Stephen G. Nash and Jorge Nocedal. 1991. A numerical study of the limited memory bfgs method and truncated-newton method for large scale optimization. *SIAM Journal on Optimization*, 1(2):358–372.

Truc-Vien T. Nguyen, Alessandro Moschitti, and Giuseppe Riccardi. 2009. Convolution kernels on constituent, dependency and sequential structures for relation extraction. In *Proceedings of EMNLP-2009*, pages 1378–1387, Singapore, August.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of English text. In *Proceedings of COLING-2004*, pages 64–70, Geneva, Switzerland, August 23-27.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan Mcdonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The conll 2007 shared task on dependency parsing. In *Proceeding of EMNLP-CoNLL 2007*, pages 915–932, Prague, Czech.

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

Mark A. Paskin. 2001. Cubic-time parsing and learning algorithms for grammatical bigram models. Technical Report, UCB/CSD-01-1148.

Slav Petrov, Dipanjan Das, and Ryan T. McDonald. 2011. A universal part-of-speech tagset. *CoRR*, abs/1104.2086.

Yusuke Shinyama, Satoshi Sekine, and Kiyoshi Sudo. 2002. Automatic paraphrase acquisition from news articles. In *Proceeding of HLT-2002*, pages 313–318.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of ACL-2005*, pages 354–362, Ann Arbor, Michigan, June.

David A. Smith and Jason Eisner. 2007. Bootstrapping feature-rich dependency parsers with entropic priors. In *Proceedings of EMNLP/CoNLL-2007*, pages 667–677, Prague, Czech Republic, June.

David A. Smith and Noah A. Smith. 2004. Bilingual parsing with factored estimation: Using English to parse Korean. In *Proceedings of EMNLP-2004*, pages 49–56.

David A. Smith and Noah A. Smith. 2007. Probabilistic models of nonporjective dependency trees. In *Proceedings of EMNLP-CONLL 2007*, pages 132–140, Prague, Czech, June.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010. From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *Proceedings of NAACL/HLT-2010*, pages 751–759, Los Angeles, California, June.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2013. Breaking out of local optima with count transforms and model recombination: A study in grammar induction. In *Proceedings of EMNLP-2013*, pages 1983–1995, Seattle, Washington, USA, October.

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

Jun Xie, Haitao Mi, and Qun Liu. 2011. A novel dependency-to-string model for statistical machine translation. In *Proceedings of EMNLP-2011*, pages 216–226, Edinburgh, Scotland, UK., July.

Hao Zhang, Liang Huang, Kai Zhao, and Ryan McDonald. 2013. Online learning for inexact hypergraph search. In *Proceedings of EMNLP-2013*, pages 908–913, Seattle, Washington, USA, October.

Unsupervised Morphology-Based Vocabulary Expansion

Mohammad Sadegh Rasooli, Thomas Lippincott, Nizar Habash and **Owen Rambow**
Center for Computational Learning Systems
Columbia University, New York, NY, USA
{rasooli,tom,habash,rambow}@ccls.columbia.edu

Abstract

We present a novel way of generating unseen words, which is useful for certain applications such as automatic speech recognition or optical character recognition in low-resource languages. We test our vocabulary generator on seven low-resource languages by measuring the decrease in out-of-vocabulary word rate on a held-out test set. The languages we study have very different morphological properties; we show how our results differ depending on the morphological complexity of the language. In our best result (on Assamese), our approach can predict 29% of the token-based out-of-vocabulary with a small amount of unlabeled training data.

1 Introduction

In many applications in human language technologies (HLT), the goal is to generate text in a target language, using its standard orthography. Typical examples include automatic speech recognition (ASR, also known as STT or speech-to-text), optical character recognition (OCR), or machine translation (MT) into a target language. We will call such HLT applications "target-language generation technologies" (TLGT). The best-performing systems for these applications today rely on training on large amounts of data: in the case of ASR, the data is aligned audio and transcription, plus large unannotated data for the language modeling; in the case of OCR, it is transcribed optical data; in the case of MT, it is aligned bitexts. More data provides for better results. For languages with rich resources, such as English, more data is often the best solution, since the required data is readily available (including bitexts), and the cost of annotating (e.g., transcribing) data is outweighed by the potential significance of the systems that the data will enable. Thus, in HLT, improvements in quality are often brought about by using larger data sets (Banko and Brill, 2001).

When we move to low-resource languages, the solution of simply using more data becomes less appealing. Unannotated data is less readily available: for example, at the time of publishing this paper, 55% of all websites are in English, the top 10 languages collectively account for 90% of web presence, and the top 36 languages have a web presence that covers at least 0.1% of web sites.[1] All other languages (and all languages considered in this paper except Persian) have a web presence of less than 0.1%. Considering Wikipedia, another resource often used in HLT, English has 4.4 million articles, while only 48 other languages have more than 100,000.[2] As attention turns to developing HLT for more languages, including low-resource languages, alternatives to "more-data" approaches become important.

At the same time, it is often not possible to use knowledge-rich approaches. For low-resource languages, resources such as morphological analyzers are not usually available, and even good scholarly descriptions of the morphology (from which a tool could be built) are often not available. The challenge is therefore to use data, but to make do with a small amount of data, and thus to use data better. This paper is a contribution to this goal. Specifically, we address TLGTs, i.e., the types of HLT mentioned above that generate target language text. We propose a new approach to generating unseen words of the target language which have not been seen in the training data. Our approach is entirely unsupervised. It assumes that word-units are specified, typically by whitespace and punctuation.

[1] http://en.wikipedia.org/wiki/Languages_used_on_the_Internet

[2] http://meta.wikimedia.org/wiki/List_of_Wikipedias

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1349–1359,
Baltimore, Maryland, USA, June 23-25 2014.

Expanding the vocabulary of the target language can be useful for TLGTs in different ways. For ASR and OCR, which can compose words from smaller units (phones or graphically recognized letters), an expanded target language vocabulary can be directly exploited without the need for changing the technology at all: the new words need to be inserted into the relevant resources (lexicon, language model) etc, with appropriately estimated probabilities. In the case of MT into morphologically rich low-resource languages, morphological segmentation is typically used in developing the translation models to reduce sparsity, but this does not guarantee against generating wrong word combinations. The expanded word combinations can be used to extend the language models used for MT to bias against incoherent hypothesized new sequences of segmented words.

Our approach relies on unsupervised morphological segmentation. We do not in this paper contribute to research in unsupervised morphological segmentation; we only use it. The contribution of this paper lies in proposing how to use the results of unsupervised morphological segmentation in order to generate unseen words of the language. We investigate several ways of doing so, and we test them on seven low-resource languages. These languages have very different morphological properties, and we show how our results differ depending on the morphological complexity of the language. In our best result (on Assamese), we show that our approach can predict 29% of the token-based out-of-vocabulary with a small amount of unlabeled training data.

The paper is structured as follows. We first discuss related work in Section 2. We then present our method in Section 3, and present experimental results in Section 4. We conclude with a discussion of future work in Section 5.

2 Related Work

Approaches to Morphological Modeling Computational morphology is a very active area of research with a multitude of approaches that vary in the degree of manual annotation needed, and the amount of machine learning used. At one extreme, we find systems that are painstakingly and carefully designed by hand (Koskenniemi, 1983; Buckwalter, 2004; Habash and Rambow, 2006; Détrez and Ranta, 2012). Next on the continuum, we find work that focuses on defining morphological models with limited lexica that are then extended using raw text (Clément et al., 2004; Forsberg et al., 2006). In the middle of this continuum, we find efforts to learn complete paradigms using fully supervised methods relying on completely annotated data points with rich morphological information (Durrett and DeNero, 2013; Eskander et al., 2013). Next, there is work on minimally supervised methods that use available resources such as dictionaries, bitexts, and other additional morphological annotations (Yarowsky and Wicentowski, 2000; Cucerzan and Yarowsky, 2002; Neuvel and Fulop, 2002; Snyder and Barzilay, 2008). At the other extreme, we find unsupervised methods that learn morphology models from unannotated data (Creutz and Lagus, 2007; Monson et al., 2008; Dreyer and Eisner, 2011; Sirts and Goldwater, 2013).

The work we present in this paper makes no use of any morphological annotations whatsoever, yet we are quite distinct from the approaches cited above. We compare our work to two efforts specifically. First, consider work in automatic morphological segmentation learning from unannotated data (Creutz and Lagus, 2007; Monson et al., 2008). Unlike these approaches which provide segmentations for training data and produce models that can be used to segment unseen words, our approach can generate words that have not been seen in the training data. The focus of efforts is rather complementary: we actually use an off-the-shelf unsupervised segmentation system (Creutz and Lagus, 2007) as part of our approach. Second, consider paradigm completion methods such as the work of Dreyer and Eisner (2011). This effort is closely related to our work although unlike it, we make no assumptions about the data and do not introduce any restrictions along the lines of derivation/inflectional morphology: Dreyer and Eisner (2011) limited their work to verbal paradigms and used annotated training data in addition to basic assumptions about the problem such as the size of the paradigms. In our approach, we have zero annotated information and we do not distinguish between inflectional and derivational morphology, nor do we limit ourselves to a specific part-of-speech (POS).

Vocabulary Expansion in HLT There have been diverse approaches towards dealing with out-of-vocabulary (OOV) words in ASR. In some models, the approach is to expand the lexicon by

adding new words or pronunciations. Ohtsuki et al. (2005) propose a two-run model where in the first run, the input speech is recognized by the reference vocabulary and relevant words are extracted from the vocabulary database and added thereafter to the reference vocabulary to build an expanded lexicon. Word recognition is done in the second run based on the lexicon. Lei et al. (2009) expanded the pronunciation lexicon via generating all possible pronunciations for a word before lattice generation and indexation. There are also other methods for generating abbreviations in voice search systems such as Yang et al. (2012). While all of these approaches involve lexicon expansion, they do not employ any morphological information.

In the context of MT, several researchers have addressed the problem of OOV words by relating them to known in-vocabulary (INV) words. Yang and Kirchhoff (2006) anticipated OOV words that are potentially morphologically related using phrase-based backoff models. Habash (2008) considered different techniques for vocabulary expansion online. One of their techniques learned models of morphological mapping between morphologically rich source words in Arabic that produce the same English translation. This was used to relate an OOV word to a morphologically related INV word. Another technique expanded the MT phrase tables with possible transliterations and spelling alternatives.

3 Morphology-based Vocabulary Expansion

3.1 Approach

Our approach to morphology-based vocabulary expansion consists of three steps (Figure 1). We start with a "training" corpus of (unannotated) words and generate a list of new (unseen) words that expands the vocabulary of the training corpus.

1. **Unsupervised Morphology Segmentation** The first step is to segment each word in the training corpus into sequences of prefixes, stem and suffixes, where the prefixes and suffixes are optional.[3]

2. **FST-based Morphology Expansion** We then construct new word models using the segmented stems and affixes. We explore two different techniques for morphology-based vocabulary expansion that we discuss below. The output of these models is represented as a weighted finite state machine (WFST).

3. **Reranking Models** Given that the size of the expanded vocabulary can be quite large and it may include a lot of over-generation, we rerank the expanded set of words before taking the top n words to use in downstream processes. We consider four reranking conditions which we describe below.

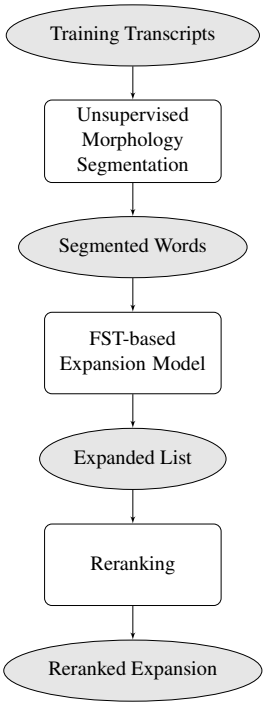


Figure 1: The flowchart of the lexicon expansion system.

3.2 Morphology Expansion Techniques

As stated above, the input to the morphology expansion step is a list of words segmented into morphemes: zero or more prefixes, one stem, and zero or more suffixes. Figure 2a presents an example of such input using English words (for clarity).

We use two different models of morphology expansion in this paper: Fixed Affix model and Bigram Affix model.

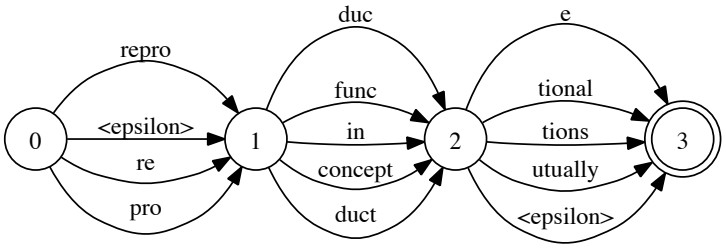
3.2.1 Fixed Affix Expansion Model

In the Fixed Affix model, we construct a set of fused prefixes from all the unique prefix sequences in the training data; and we similarly construct a

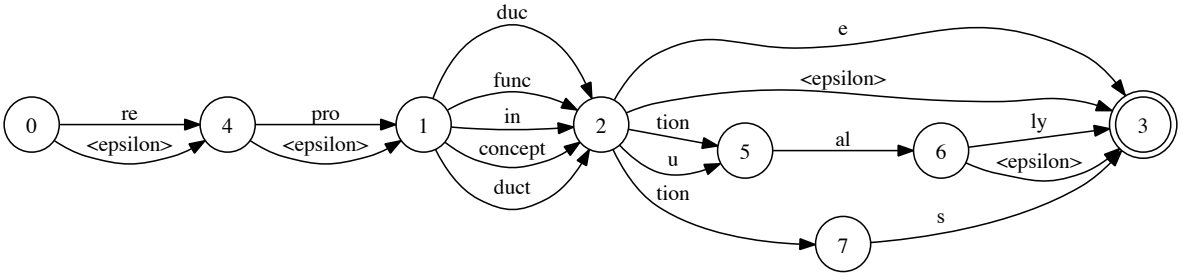
[3] In this paper, we use an off-the-shelf system for this step but plan to explore new methods in the future, such as joint segmentation and expansion.

re+ pro+ duc +e
func +tion +al
re+ duc +e
re+ duc +tion +s
in
pro+ duct
concept +u +al + ly

(a) Training data with morpheme boundaries. Prefixes end with and suffixes start with "+" signs.



(b) FST for the Fixed Affix expansion model



(c) FST for the Bigram Affix expansion model

Figure 2: Two models of word generation from morphologically annotated data. In our experiments, we used weighted finite state machine. We use character-based WFST in the implementation to facilitate analyzing inputs as well as word generation.

set of fused suffixes from all the unique suffix sequences in the training data. In other words, we simply pick characters from beginning of the word up to the first stem as the prefix and characters from the first suffix to the end of the word as the suffix. Everything in the middle is the stem. In this model, each word has one single prefix and one single suffix (each of which can be empty independently). The Fixed Affix model is simply the concatenation of the disjunction of all prefixes with the disjunction of all stems and the disjunction of all suffixes into one FST:

$$\textit{prefix} \rightarrow \textit{stem} \rightarrow \textit{suffix}$$

The morpheme paths in the FST are weighted to reflect their probability in the training corpus.[4] Figure 2b exemplifies a Fixed Affix model derived from the example training data in Figure 2a.

[4]We convert the probability into a cost by taking the negative of the log of the probability.

3.2.2 Bigram Affix Expansion Model

In the Bigram Affix model, we do the same for the stem as in the Fixed Affix model, but for prefixes and suffixes, we create a bigram language model in the finite state machine. The advantage of this technique is that unseen compound affixes can be generated by our model. For example, the Fixed Affix model in Figure 2b cannot generate the word *func+tion+al+ly* since the suffix *+tionally* is not seen in the training data. However, this word can be generated in the Bigram Affix model as shown in Figure 2c: there is a path passing $0 \rightarrow 4 \rightarrow 1 \rightarrow 2 \rightarrow 5 \rightarrow 6 \rightarrow 3$ in the FST that can produce this word. We expect this model to have better recall for generating new words in the language because of its affixation flexibility.

3.3 Reranking Techniques

The expanded models allow for a large number of words to be generated. We limit the number of vocabulary expansion using different thresholds after reranking or reweighing the WFSTs generated

above. We consider four reranking conditions.

3.3.1 No Reranking (NRR)

The baseline reranking option is no reranking (NRR). In this approach we use the weights in the WFST, which are based on the independent prefix/stem/suffix probabilities, to determine the ranking of the expanded vocabulary.

3.3.2 Trigraph-based Reweighting (W∘Tr)

We reweight the weights in the WFST model (Fixed or Bigram) by composing it with a *letter trigraph language model* ($W \circ \text{Tr}$). A letter trigraph LM is itself a WFST where each trigraph (a sequence of three consequent letters) has an associated weight equal to its negative log-likelihood in the training data. This reweighting allows us to model preferences of sequences of word letters seen more in the training data. For example, in a word like *producttions*, the trigraphs *ctt* and *tti* are very rare and thus decrease its probability.

3.3.3 Trigraph-based Reranking (TRR)

When we compose our initial WFST with the trigraph FST, the probability of each generated word from the new FST is equal to the product of the probability of its morphemes and the probabilities of each trigraph in that word. This basically makes the model prefer shorter words and may degrade the effect of morphology information. Instead of reweighting the WFST, we get the n-best list of generated words and rerank them using their trigraph probabilities. We will refer to this technique as TRR.

3.3.4 Reranking Morpheme Boundaries (BRR)

The last reranking technique reranks the n-best generated word list with trigraphs that are incident on the morpheme boundaries (in case of Bigram Affix model, the last prefix and first suffix). The intuition is that we already know that any morpheme that is generated from the morphology FST is already seen in the training data but the boundary for different morphemes are not guaranteed to be seen in the training data. For example, for the word *producttions*, we only take into account the trigraphs *rod*, *odu*, *ctt* and *tti* instead of all possible trigraphs. We will refer to this technique as BRR.

4 Evaluation

4.1 Evaluation Data and Tools

Evaluation Data The IARPA Babel program is a research program for developing rapid spoken detection systems for under-resourced languages (Harper, 2013). We use the IARPA Babel program limited language pack data which consists of 20 hours of telephone speech with transcription. We use six languages which are known to have rich morphology: Assamese (IARPA-babel102b-v0.5a), Bengali (IARPA-babel103b-v0.4b), Pashto (IARPA-babel104b-v0.4bY), Tagalog (IARPA-babel106-v0.2g), Turkish (IARPA-babel105b-v0.4) and Zulu (IARPA-babel206b-v0.1e). Speech annotation such as silences and hesitations are removed from transcription and all words are turned into lower-case (for languages using the Roman script – Tagalog, Turkish and Zulu). Moreover, in order to be able to perform a manual error analysis, we include a language that has rich morphology and of which the first author is a native speaker: Persian. We sampled data from the training and development set of the Persian dependency treebank (Rasooli et al., 2013) to create a comparable seventh dataset in Persian. Statistics about the datasets are shown in Table 1. We also conduct further experiments on just verbs and nouns in the data set for Persian (Persian-N and Persian V). As shown in Table 1, the training data is very small and the OOV rate is high especially in terms of types. For some languages that have richer morphology such as Turkish and Zulu, the OOV rate is much higher than other languages.

Word Generation Tools and Settings For unsupervised learning of morphology, we use Morfessor CAT-MAP (v. 0.9.2) which was shown to be a very accurate morphological analyzer for morphologically rich languages (Creutz and Lagus, 2007). In order to be able to analyze Unicode-based data, we convert each character in our dataset to some conventional ASCII character and then train Morfessor on the mapped dataset; after finishing the training, we map the data back to the original character set. We use the default setting in Morfessor for unsupervised learning.

For preparing the WFST, we use OpenFST (Riley et al., 2009). We get the top one million shortest paths (i.e., least costly paths of words) and apply our reranking models on them. It is worth pointing out that our WFSTs are character-based

Language	Training Data		Development Data			
	Type	Token	Type	Token	Type OOV%	Token OOV%
Assamese	8694	73151	7253	66184	49.57	8.28
Bengali	9460	81476	7794	70633	50.65	8.47
Pashto	6968	115069	6135	108137	44.89	4.25
Persian	14047	71527	10479	42939	44.16	12.78
Tagalog	6213	69577	5480	64334	54.95	7.81
Turkish	11985	77128	9852	67042	56.84	12.34
Zulu	15868	65655	13756	57141	68.72	21.76
Persian-N	9204	31369	7502	18816	46.36	22.11
Persian-V	2653	11409	1332	7318	41.07	9.01

Table 1: Statistics of training and development data for morphology-based unsupervised word generation experiments.

and thus we also have a morphological analyzer that can give all possible segmentations for a given word. By running the morphological analyzer on the OOVs, we can have the potential upper bound of OOV reduction by the system (labeled "∞" in Tables 2 and 3).

4.2 Lexicon Expansion Results

The results for lexicon expansion are shown in Table 2 for types and Table 3 for tokens.

We use the trigraph WFST as our baseline model. This model does not use any morphological information. In this case, words are generated according to the likelihood of their trigraphs, without using any information from the morphological segmentation. We call this model the trigraph WFST (Tr. WFST). We consistently have better numbers than this baseline in all of our models except for Pashto when measured by tokens. ∞ is the upper-bound OOV reduction for our expansion model: for each word in the development set, we ask if our model, without any vocabulary size restriction at all, could generate it.

The best results (again, except for Pashto) are achieved using one of the three reranking methods (reranking by trigraph probabilities or morpheme boundaries) as opposed to doing no reranking. To our surprise, the Fixed Affix model does a slightly better job in reducing out of vocabulary than the Bigram Affix model. We can also see from the results that reranking in general is very effective.

We also compare our models with the case that there is much more training data and we do not do vocabulary expansion at all. In Table 2 and Table 3, "FP" indicates the full language pack for the Babel project data which is approximately six to eight times larger than the limited pack training data, and the full training data for Persian which is approximately five times larger. We see that the larger training data outperforms our methods in all languages. However, from the results of ∞, which is the upper-bound OOV reduction by our expansion model, for some languages such as Assamese, our numbers are close to the FP results and for Zulu it is even better than FP.

We also study how OOV reduction is affected by the size of the generated vocabulary. The trends for different sizes of the lexicon expansion by Fixed Affix model that is reranked by trigraph probabilities is shown in Figure 3. As seen in the results, for languages that have richer morphology, it is harder to achieve results near to the upper bound. As an outlier, morphology does not help for Pashto. One possible reason might be that based on the results in Table 4, Morfessor does not explore morphology in Pashto as well as other languages.

Morphological Complexity As for further analysis, we can study the correlation between morphological complexity and hardness of reducing OOVs. Much work has been done in linguistics to classify languages (Sapir, 1921; Greenberg, 1960). The common wisdom is that languages are not either agglutinative or fusional, but are on a spectrum; however, no work to our knowledge places all languages (or at least the ones we worked on) on such a spectrum. We propose several metrics. First, we can consider the number of unique affixival morphemes in each language, as determined by Morfessor. As shown in Table 4 ($|pr| + |sf|$), Zulu has the most morphemes and Pashto the fewest. A second possible metric of the

Language	Tr. WFST	Fixed Affix Model					Bigram Affix Model					FP
		NRR	W∘Tr	TRR	BRR	∞	NRR	W∘Tr	TRR	BRR	∞	
Assamese	15.94	24.03	**28.46**	28.15	27.15	48.07	23.50	28.15	27.84	26.59	51.02	50.96
Bengali	15.68	20.09	**24.75**	24.49	22.54	40.98	21.78	24.65	24.67	23.51	42.55	48.83
Pashto	18.70	19.03	19.28	19.24	18.63	25.13	**19.43**	18.81	18.92	18.77	25.24	64.96
Persian	12.83	18.95	18.39	19.30	**19.99**	50.11	18.58	18.09	18.65	18.84	53.13	58.45
Tagalog	11.39	14.61	16.51	16.21	**16.81**	35.64	14.45	16.01	15.81	16.74	38.72	53.64
Turkish	07.75	09.11	**14.79**	**14.79**	14.71	55.48	09.04	13.63	14.34	13.52	66.54	53.54
Zulu	07.63	11.87	12.96	**13.87**	13.68	66.73	12.04	12.35	13.69	13.75	82.38	35.62
Average	12.85	16.81	**19.31**	**19.31**	19.07	46.02	17.02	18.81	19.13	18.81	51.37	52.29
Persian-N	14.86	**24.67**	22.74	22.83	24.15	37.32	23.78	21.68	22.51	23.32	38.38	-
Persian-V	54.84	68.19	72.39	**73.49**	71.12	80.44	67.28	71.48	72.58	70.02	80.62	-

Table 2: Type-based expansion results for the 50k-best list for different models. Tr. WFST stands for trigraph WFST, NRR for no reranking, W∘Tr for trigraph reweighting, TRR for trigraph-based rereanking, BRR for reranking morpheme boundary, and ∞ for the upper bound of OOV reduction via lexicon expansion if we produce all words. FP (full-pack data) shows the effect of using bigger data with the size of about seven times larger than our data set, instead of using our unsupervised approach.

Language	Tr. WFST	Fixed Affix Model					Bigram Affix Model					FP
		NRR	W∘Tr	TRR	BRR	∞	NRR	W∘Tr	TRR	BRR	∞	
Assamese	18.07	25.70	**29.43**	29.12	28.13	47.88	25.34	29.06	28.82	27.64	50.31	58.03
Bengali	17.79	20.91	**25.61**	25.27	23.65	40.60	22.58	25.20	25.41	24.77	42.22	55.92
Pashto	**21.27**	19.40	19.94	19.92	18.59	25.45	19.68	19.40	19.29	18.72	25.58	71.46
Persian	14.78	20.77	20.32	21.30	**22.03**	51.00	20.63	19.72	20.61	20.95	54.01	63.10
Tagalog	12.88	14.55	**16.88**	16.36	16.60	33.95	14.37	16.12	16.12	16.38	37.07	61.53
Turkish	09.97	11.42	**17.82**	17.67	17.23	56.54	11.05	16.82	17.41	15.98	66.54	59.68
Zulu	08.85	13.70	14.72	15.62	**15.67**	68.07	13.70	14.07	15.47	15.60	87.90	41.27
Average	14.80	18.06	20.67	**20.75**	20.27	44.78	18.19	20.48	20.45	20.01	51.95	58.71
Persian-N	16.82	**26.46**	24.42	24.56	25.71	38.40	25.69	23.50	24.20	25.04	39.41	–
Persian-V	60.09	71.47	75.57	**76.48**	73.60	82.55	70.56	74.81	75.72	72.53	82.70	–

Table 3: Token-based expansion results for the 50k-best list for different models. Abbreviations are the same as Table 2.

complexity of the morphology is by calculating the average number of unique prefix-suffix pairs in the training data after morpheme segmentation which is shown as $|If|$ in Table 4. Finally, a third possible metric is the number of all possible words that can be generated ($|L|$). These three metrics correlate fairly well across the languages.

The metrics we propose also correlate with commonly accepted classifications: e.g., Zulu and Turkish (highly agglutinative) have higher scores in terms of our $|pr| + |sf|$, $|If|$ and $|L|$ metrics in Table 4 than other languages. The results from full language packs in Table 3 also show that there is a reverse interaction of morphological complexity and the effect of blindly adding more data. Thus for morphologically rich languages, adding more data is less effective than for languages with poor morphology.

The size of the languages ($|L|$) suggests that we are suffering from vast overgeneration; we overgenerate because in our model any affix can attach to any stem, which is not in general true. Thus there is a lack of linguistic knowledge such as paradigm information (Stump, 2001) for each word category in our model. In other words, all morphemes are treated the same in our model which is not true in natural languages. One way to tackle this problem is through an unsupervised POS tagger. The challenge here is that fully unsupervised POS taggers (without any tag dictionary) are not very accurate (Christodoulopoulos et al., 2010). Another way is through using joint mor-

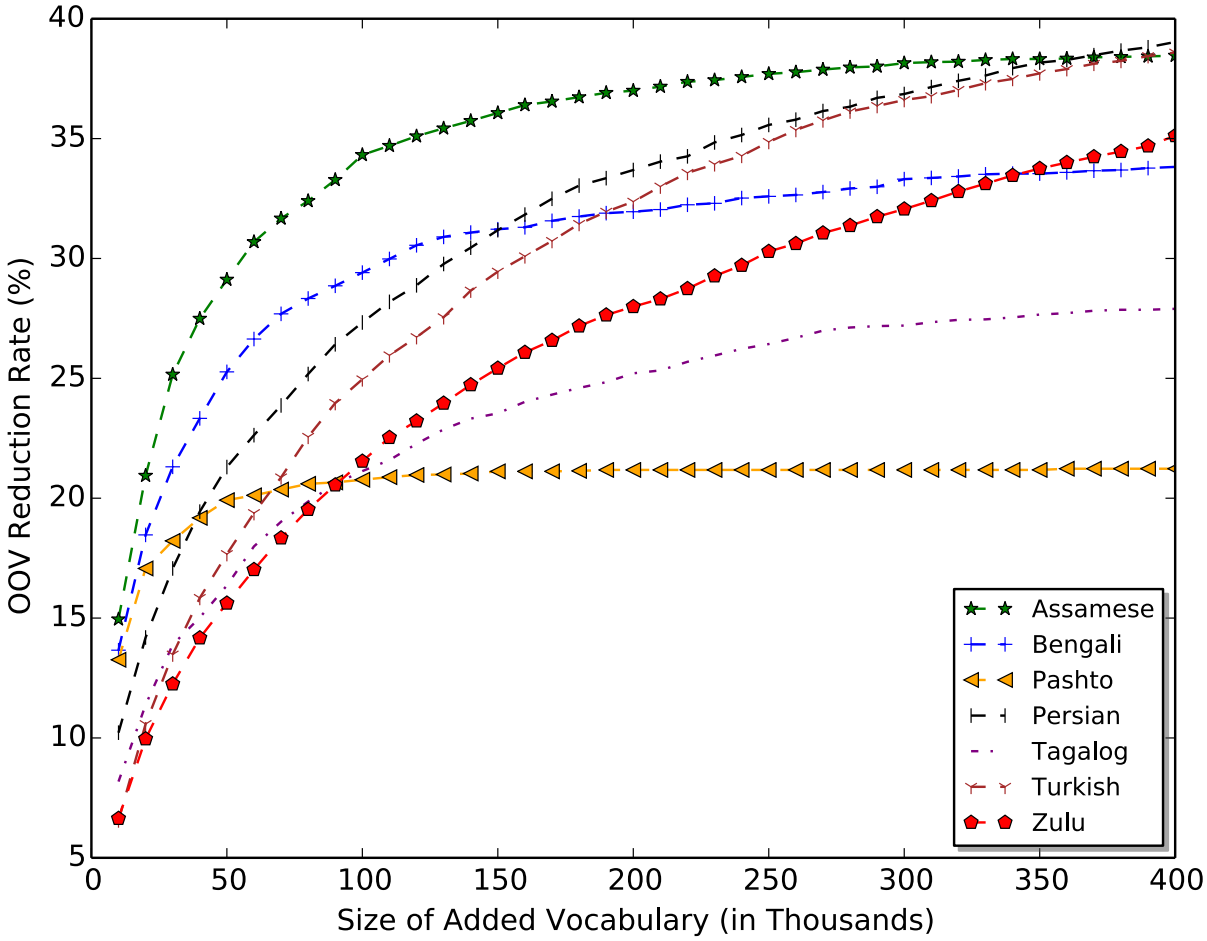


Figure 3: Trends for token-based OOV reduction with different sizes for the Fixed Affix model with trigraph reranking.

Language	$\|pr\|$	$\|stm\|$	$\|sf\|$	$\|L\|$	$\|If\|$
Assamese	4	4791	564	10.8M	1.8
Bengali	3	6496	378	7.4M	1.5
Pashto	1	5395	271	1.5M	1.3
Persian	49	6998	538	184M	2.0
Tagalog	179	4259	299	228M	1.5
Turkish	45	5266	1801	427M	2.3
Zulu	2254	5680	427	5.5B	2.8
Persian-N	3	6121	268	4.9M	1.5
Persian-V	43	788	44	1.5M	3.4

Table 4: Information about the number of unique morphemes in the Fixed Affix model for each dataset including empty affixes. $|L|$ shows the upper bound of the number of possible unique words that can be generated from the word generation model. $|If|$ is the average number of unique prefix-suffix pairs (including empty pairs) for each stem.

phology and tagging models such as Frank et al. (2013).

Error Analysis on Turkish Unfortunately for most languages we could not find an available rule-based or supervised morphological analyzer to verify the words generated by our model. The only available tool for us is a Turkish finite-state morphological analyzer (Oflazer, 1996) implemented with the Xerox FST toolkit (Beesley and Karttunen, 2003). As we can see in Table 5, the system with the largest proportion of correct generated words reranks the expansion with trigraph probabilities using a Fixed Affix model. Results also show that we are overgenerating many nonsense words that we ought to be pruning from our results. Another observation is that the recognition percentage of the morphological analyzer on INV words is much higher than on OOVs, which shows that OOVs in Turkish dataset are much harder to analyze.

Model		Precision
Tr. WFST		17.19
Fixed Affix Model	NRR	13.36
	W∘Tr	25.66
	TRR	**26.30**
	BRR	25.14
Bigram Affix Model	NRR	12.94
	W∘Tr	24.21
	TRR	25.39
	BRR	23.45
Development	words	89.30
	INVs	95.44
	OOVs	84.64

Table 5: Results from running a hand-crafted Turkish morphological analyzer (Oflazer, 1996) on different expansions and on the development set. Precision refers to the percentage of the words are recognized by the analyzer. The results on development are also separated into INV and OOV.

Error Analysis on Persian From the best 50k word result for Persian (Fixed Affix Model:BRR), we randomly picked 200 words and manually analyzed them. 89 words are correct (45.5%) where 55.0% of these words are from noun affixation, 23.6% from verb clitics, 9.0% from verb inflections, 5.6% from incorrect affixations that accidentally resulted in possible words, 4.5% from uninflected stems, and a few from adjective affixation. Among incorrectly generated words, 65.8% are from combining a stem of one POS with affixes from another POS (e.g., attaching a noun affix to a verb stem), 14.4% from combining a stem with affixes which are compatible with POS but not allowed for that particular stem (e.g., there is a noun suffix that can only attach to a subset of noun stems), 9.0% are from wrong affixes produced by Morfessor and others are from incorrect vowel harmony or double affixation.

In order to study the effect of vocabulary expansion more deeply, we trained a subset of all nouns and verbs in the same dataset (also shown in Table 1). Verbs in Persian have rich but more or less regular morphology, while nouns, which have many irregular cases, have rich morphology but not as rich as verbs. The results in Table 4 show that Morfessor captures these phenomena. Furthermore, our results in Table 2 and Table 3 show that our performance on OOV reduction with verbs is far superior to our performance with nouns. We also randomly picked 200 words from each of the experiments (noun and verbs) to study the degree of correctness of generated forms. For nouns, 94 words are correct and for verbs only 71 words are correct. Most verb errors are due to incorrect morpheme extraction by Morfessor. In contrast, most noun errors result from affixes that are only compatible with a subset of all possible noun stems. This suggests that if we conduct experiments using more accurate unsupervised morphology and also have a more fine-grained paradigm completion model, we might improve our performance.

5 Conclusion and Future Work

We have presented an approach to generating new words. This approach is useful for low-resource, morphologically rich languages. It provides words that can be used in HLT applications that require target-language generation in this language, such as ASR, OCR, and MT. An implementation of our approach, named BabelGUM (Babel General Unsupervised Morphology), will be publicly available. Please contact the authors for more information.

In future work we will explore the possibility of jointly performing unsupervised morphological segmentation with clustering of words into classes with similar morphological behavior. These classes will extend POS classes. We will tune the system for our purposes, namely OOV reduction.

Acknowledgements

We thank Anahita Bhiwandiwalla, Brian Kingsbury, Lidia Mangu, Michael Picheny, Benoît Sagot, Murat Saraclar, and Géraldine Walther for helpful discussions. The project is supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense U.S. Army Research Laboratory (DoD/ARL) contract number W911NF-12-C-0012. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government.

References

Michele Banko and Eric Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, pages 26–33, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kenneth R Beesley and Lauri Karttunen. 2003. Finite-state morphology: Xerox tools and techniques. *CSLI, Stanford*.

Tim Buckwalter. 2004. Buckwalter Arabic Morphological Analyzer Version 2.0. LDC catalog number LDC2004L02, ISBN 1-58563-324-0.

Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. 2010. Two decades of unsupervised pos induction: How far have we come? In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 575–584. Association for Computational Linguistics.

Lionel Clément, Benoît Sagot, and Bernard Lang. 2004. Morphology based automatic acquisition of large-coverage lexica. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*. European Language Resources Association (ELRA).

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing (TSLP)*, 4(1):3.

Silviu Cucerzan and David Yarowsky. 2002. Bootstrapping a multilingual part-of-speech tagger in one person-day. In *The 6th Conference on Natural Language Learning (CoNLL-2002)*, pages 1–7.

Grégoire Détrez and Aarne Ranta. 2012. Smart paradigms and the predictability and complexity of inflectional morphology. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 645–653. Association for Computational Linguistics.

Markus Dreyer and Jason Eisner. 2011. Discovering morphological paradigms from plain text using a dirichlet process mixture model. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 616–627. Association for Computational Linguistics.

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1185–1195. Association for Computational Linguistics.

Ramy Eskander, Nizar Habash, and Owen Rambow. 2013. Automatic extraction of morphological lexicons from morphologically annotated corpora. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1032–1043, Seattle, Washington, USA, October. Association for Computational Linguistics.

Markus Forsberg, Harald Hammarström, and Aarne Ranta. 2006. Morphological lexicon extraction from raw text data. *Advances in Natural Language Processing*, pages 488–499.

Stella Frank, Frank Keller, and Sharon Goldwater. 2013. Exploring the utility of joint morphological and syntactic learning from child-directed speech. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 30–41. Association for Computational Linguistics.

Joseph H Greenberg. 1960. A quantitative approach to the morphological typology of language. *International journal of American linguistics*, pages 178–194.

Nizar Habash and Owen Rambow. 2006. MAGEAD: A morphological analyzer and generator for the Arabic dialects. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 681–688, Sydney, Australia.

Nizar Habash. 2008. Four techniques for online handling of out-of-vocabulary words in Arabic-English statistical machine translation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 57–60. Association for Computational Linguistics.

Mary Harper. 2013. The babel program and low resource speech technology. In *Automatic Speech Recognition and Understanding Workshop (ASRU) Invited talk*.

Kimmo Koskenniemi. 1983. Two-Level Model for Morphological Analysis. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 683–685.

Xin Lei, Wen Wang, and Andreas Stolcke. 2009. Data-driven lexicon expansion for Mandarin broadcast news and conversation speech recognition. In *International conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4329–4332.

Christian Monson, Jaime Carbonell, Alon Lavie, and Lori Levin. 2008. Paramor: Finding paradigms across morphology. *Advances in Multilingual and Multimodal Information Retrieval*, pages 900–907.

Sylvain Neuvel and Sean A Fulop. 2002. Unsupervised learning of morphology without morphemes. In *Proceedings of the ACL-02 workshop on Morphological and phonological learning-Volume 6*, pages 31–40. Association for Computational Linguistics.

Kemal Oflazer. 1996. Error-tolerant finite-state recognition with applications to morphological analysis and spelling correction. *Computational Linguistics*, 22(1):73–89.

Katsutoshi Ohtsuki, Nobuaki Hiroshima, Masahiro Oku, and Akihiro Imamura. 2005. Unsupervised vocabulary expansion for automatic transcription of broadcast news. In *International conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1021–1024.

Mohammad Sadegh Rasooli, Manouchehr Kouhestani, and Amirsaeid Moloodi. 2013. Development of a Persian syntactic dependency treebank. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 306–314. Association for Computational Linguistics.

Michael Riley, Cyril Allauzen, and Martin Jansche. 2009. Openfst: An open-source, weighted finite-state transducer library and its applications to speech and language. In *Human Language Technologies Tutorials: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 9–10.

Edward Sapir. 1921. *Language: An introduction to the study of speech*. Harcourt, Brace and company (New York).

Kairit Sirts and Sharon Goldwater. 2013. Minimally-supervised morphological segmentation using adaptor grammars. *Transactions for the ACL*, 1:255–266.

Benjamin Snyder and Regina Barzilay. 2008. Unsupervised multilingual learning for morphological segmentation. In *Proceedings of the 46th annual meeting of the association for computational linguistics: Human language Technologies (ACL-HLT)*, pages 737–745. Association for Computational Linguistics.

Gregory T. Stump. 2001. *A theory of paradigm structure*. Cambridge.

Mei Yang and Katrin Kirchhoff. 2006. Phrase-based backoff models for machine translation of highly inflected languages. In *Proceedings of Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 41–48, Trento, Italy.

Dong Yang, Yi-Cheng Pan, and Sadaoki Furui. 2012. Vocabulary expansion through automatic abbreviation generation for Chinese voice search. *Computer Speech & Language*, 26(5):321–335.

David Yarowsky and Richard Wicentowski. 2000. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 207–216.

Toward Better Chinese Word Segmentation for SMT via Bilingual Constraints

Xiaodong Zeng[†] **Lidia S. Chao**[†] **Derek F. Wong**[†] **Isabel Trancoso**[‡] **Liang Tian**[†]
[†]NLP[2]CT Lab / Department of Computer and Information Science, University of Macau
[‡]INESC-ID / Instituto Superior Ténico, Lisboa, Portugal
nlp2ct.samuel@gmail.com, {lidiasc, derekfw}@umac.mo,
isabel.trancoso@inesc-id.pt,tianliang0123@gmail.com

Abstract

This study investigates on building a better Chinese word segmentation model for statistical machine translation. It aims at leveraging word boundary information, automatically learned by bilingual character-based alignments, to induce a preferable segmentation model. We propose dealing with the induced word boundaries as soft constraints to bias the continuous learning of a supervised CRFs model, trained by the treebank data (labeled), on the bilingual data (unlabeled). The induced word boundary information is encoded as a graph propagation constraint. The constrained model induction is accomplished by using posterior regularization algorithm. The experiments on a Chinese-to-English machine translation task reveal that the proposed model can bring positive segmentation effects to translation quality.

1 Introduction

Word segmentation is regarded as a critical procedure for high-level Chinese language processing tasks, since Chinese scripts are written in continuous characters without explicit word boundaries (e.g., space in English). The empirical works show that word segmentation can be beneficial to Chinese-to-English statistical machine translation (SMT) (Xu et al., 2005; Chang et al., 2008; Zhao et al., 2013). In fact most current SMT models assume that parallel bilingual sentences should be segmented into sequences of tokens that are meant to be "words" (Ma and Way, 2009). The practice in state-of-the-art MT systems is that Chinese sentences are tokenized by a monolingual supervised word segmentation model trained on the hand-annotated treebank data, e.g., Chinese treebank (CTB) (Xue et al., 2005). These models are conducive to MT to some extent, since they commonly have relatively good aggregate performance and segmentation consistency (Chang et al., 2008). But one outstanding problem is that these models may leave out some crucial segmentation features for SMT, since the output words conform to the treebank segmentation standard designed for monolingually linguistic intuition, rather than specific to the SMT task.

In recent years, a number of works (Xu et al., 2005; Chang et al., 2008; Ma and Way, 2009; Xi et al., 2012) attempted to build segmentation models for SMT based on bilingual unsegmented data, instead of monolingual segmented data. They proposed to learn gainful bilingual knowledge as golden-standard segmentation supervisions for training a bilingual unsupervised model. Frequently, the bilingual knowledge refers to the mappings of an individual English word to one or more consecutive Chinese characters, generated via statistical character-based alignment. They leverage such mappings to either constitute a Chinese word dictionary for maximum-matching segmentation (Xu et al., 2004), or form labeled data for training a sequence labeling model (Paul et al., 2011). The prior works showed that these models help to find some segmentations tailored for SMT, since the bilingual word occurrence feature can be captured by the character-based alignment (Och and Ney, 2003). However, these models tend to miss out other linguistic segmentation patterns as monolingual supervised models, and suffer from the negative effects of erroneously alignments to word segmentation.

This paper proposes an alternative Chinese Word Segmentation (CWS) model adapted to the SMT task, which seeks not only to maintain the advantages of a monolingual supervised model, having hand-annotated linguistic knowledge, but also to assimilate the relevant bilingual segmenta-

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1360–1369,
Baltimore, Maryland, USA, June 23-25 2014.

tion nature. We propose leveraging the bilingual knowledge to form learning constraints that guide a supervised segmentation model toward a better solution for SMT. Besides the bilingual motivated models, character-based alignment is also employed to achieve the mappings of the successive Chinese characters and the target language words. Instead of directly merging the characters into concrete segmentations, this work attempts to extract word boundary distributions for character-level trigrams (types) from the "chars-to-word" mappings. Furthermore, these word boundaries are encoded into a graph propagation (GP) expression, in order to widen the influence of the induced bilingual knowledge among Chinese texts. The GP expression constrains similar types having approximated word boundary distributions. Crucially, the GP expression with the bilingual knowledge is then used as side information to regularize a CRFs (conditional random fields) model's learning over treebank and bitext data, based on the posterior regularization (PR) framework (Ganchev et al., 2010). This constrained learning amounts to a jointly coupling of GP and CRFs, i.e., integrating GP into the estimation of a parametric structural model.

This paper is structured as follows: Section 2 points out the main differences with the related works of this study. Section 3 presents the details of the proposed segmentation model. Section 4 reports the experimental results of the proposed model for a Chinese-to-English MT task. The conclusion is drawn in Section 5.

2 Related Work

In the literature, many approaches have been proposed to learn CWS models for SMT. They can be put into two categories, monolingual-motivated and bilingual-motivated. The former primarily optimizes monolingual supervised models according to some predefined segmentation properties that are manually summarized from empirical MT evaluations. Chang et al. (2008) enhanced a CRFs segmentation model in MT tasks by tuning the word granularity and improving the segmentation consistence. Zhang et al. (2008) produced a better segmentation model for SMT by concatenating various corpora regardless of their different specifications. Distinct from their behaviors, this work uses automatically learned constraints instead of manually defined ones. Most importantly, the constraints have a better learning guidance since they originate from the bilingual texts. On the other hand, the bilingual-motivated CWS models typically rely on character-based alignments to generate segmentation supervisions. Xu et al. (2004) proposed to employ "chars-to-word" alignments to generate a word dictionary for maximum matching segmentation in SMT task. The works in (Ma and Way, 2009; Zhao et al., 2013) extended the dictionary extraction strategy. Ma and Way (2009) adopted co-occurrence frequency metric to iteratively optimize "candidate words" extract from the alignments. Zhao et al. (2013) attempted to find an optimal subset of the dictionary learned by the character-based alignment to maximize the MT performance. Paul et al. (2011) used the words learned from "chars-to-word" alignments to train a maximum entropy segmentation model. Rather than playing the "hard" uses of the bilingual segmentation knowledge, i.e., directly merging "char-to-word" alignments to words as supervisions, this study extracts word boundary information of characters from the alignments as soft constraints to regularize a CRFs model's learning.

The graph propagation (GP) technique provides a natural way to represent data in a variety of target domains (Belkin et al., 2006). In this technique, the constructed graph has vertices consisting of labeled and unlabeled examples. Pairs of vertices are connected by weighted edges encoding the degree to which they are expected to have the same label (Zhu et al., 2003). Many recent works, such as by Subramanya et al. (2010), Das and Petrov (2011), Zeng et al. (2013; 2014) and Zhu et al. (2014), proposed GP for inferring the label information of unlabeled data, and then leverage these GP outcomes to learn a semi-supervised scalable model (e.g., CRFs). These approaches are referred to as pipelined learning with GP. This study also works with a similarity graph, encoding the learned bilingual knowledge. But, unlike the prior pipelined approaches, this study performs a joint learning behavior in which GP is used as a learning constraint to interact with the CRFs model estimation.

One of our main objectives is to bias CRFs model's learning on unlabeled data, under a non-linear GP constraint encoding the bilingual knowledge. This is accomplished by the posterior regularization (PR) framework (Ganchev et

al., 2010). PR performs regularization on posteriors, so that the learned model itself remains simple and tractable, while during learning it is driven to obey the constraints through setting appropriate parameters. The closest prior study is constrained learning, or learning with prior knowledge. Chang et al. (2008) described constraint driven learning (CODL) that augments model learning on unlabeled data by adding a cost for violating expectations of constraint features designed by domain knowledge. Mann and McCallum (2008) and McCallum et al. (2007) proposed to employ generalized expectation criteria (GE) to specify preferences about model expectations in the form of linear constraints on some feature expectations.

3 Methodology

This work aims at building a CWS model adapted to the SMT task. The model induction is shown in Algorithm 1. The input data requires two types of training resources, segmented Chinese sentences from treebank $\mathcal{D}_l^c$ and parallel unsegmented sentences of Chinese and foreign language $\mathcal{D}_u^c$ and $\mathcal{D}_u^f$. The first step is to conduct character-based alignment over bitexts $\mathcal{D}_u^c$ and $\mathcal{D}_u^f$, where every Chinese character is an alignment target. Here, we are interested on n-to-1 alignment patterns, i.e., one target word is aligned to one or more source Chinese characters. The second step aims to collect word boundary distributions for all types, i.e., character-level trigrams, according to the n-to-1 mappings (Section 3.1). The third step is to encode the induced word boundary information into a k-nearest-neighbors (k-NN) similarity graph constructed over the entire set of types from $\mathcal{D}_l^c$ and $\mathcal{D}_u^c$ (Section 3.2). The final step trains a discriminative sequential labeling model, conditional random fields, on $\mathcal{D}_l^c$ and $\mathcal{D}_u^c$ under bilingual constraints in a graph propagation expression (Section 3.3). This constrained learning is carried out based on posterior regularization (PR) framework (Ganchev et al., 2010).

Algorithm 1 CWS model induction with bilingual constraints

Require:
Segmented Chinese sentences from treebank $\mathcal{D}_l^c$; Parallel sentences of Chinese and foreign language $\mathcal{D}_u^c$ and $\mathcal{D}_u^f$

Ensure:
θ: the CRFs model parameters

1: $\mathcal{D}^{c\leftrightarrow f} \leftarrow$ **char_align_bitext** $(\mathcal{D}_u^c, \mathcal{D}_u^f)$
2: $r \leftarrow$ **learn_word_bound** $(\mathcal{D}^{c\leftrightarrow f})$
3: $\mathcal{G} \leftarrow$ **encode_graph_constraint** $(\mathcal{D}_l^c, \mathcal{D}_u^c, r)$
4: $\theta \leftarrow$ **pr_crf_graph** $(\mathcal{D}_l^c, \mathcal{D}_u^c, \mathcal{G})$

3.1 Word Boundaries Learned from Character-based Alignments

The gainful supervisions toward a better segmentation solution for SMT are naturally extracted from MT training resources, i.e., bilingual parallel data. This study employs an approximated method introduced in (Xu et al., 2004; Ma and Way, 2009; Chung and Gildea, 2009) to learn bilingual segmentation knowledge. This relies on statistical character-based alignment: first, every Chinese character in the bitexts is divided by a white space so that individual characters are regarded as special "words" or alignment targets, and second, they are connected with English words by using a statistical word aligner, e.g., GIZA++ (Och and Ney, 2003). Note that the aligner is restricted to use an n-to-1 alignment pattern. The primary idea is that consecutive Chinese characters are grouped to a candidate word, if they are aligned to the same foreign word. It is worth mentioning that prior works presented a straightforward usage for candidate words, treating them as golden segmentations, either dictionary units or labeled resources. But this study treats the induced candidate words in a different way. We propose to extract the word boundary distributions[1] for character-level trigrams (*type*)[2], as shown in Figure 1, instead of the very specific words. There are two main reasons to do so. First, it is a more general expression which can reduce the impact amplification of erroneous character alignments. Second, boundary distributions can play more flexible roles as constraints over labelings to bias the model learning.

The type-level word boundary extraction is formally described as follows. Given the ith sentence pair $\langle x_i^c, x_i^f, \mathcal{A}_i^{c\rightarrow f}\rangle$ of the aligned bilingual corpus $\mathcal{D}^{c\leftrightarrow f}$, the Chinese sentence x_i^c consisting of m characters $\{x_{i,1}^c, x_{i,2}^c, ..., x_{i,m}^c\}$, and the foreign language sentence x_i^f, consisting of

[1] The distribution is on four word boundary labels indicating the character positions in a word, i.e., **B** (begin), **M** (middle), **E** (end) and **S** (single character).

[2] A word boundary distribution corresponds to the center character of a type. In fact, it aims at reducing label ambiguities to collect boundary information of character trigrams, rather than individual characters (Altun et al., 2006).

n words $\{x^f_{i,1}, x^f_{i,2}, ..., x^f_{i,n}\}$, $\mathcal{A}^{c\to f}_i$ represents a set of alignment pairs $a_j = \langle C_j, x^f_{i,j}\rangle$ that defines connections between a few Chinese characters $C_j = \{x^c_{i,j_1}, x^c_{i,j_2}, ..., x^c_{i,j_k}\}$ and a single foreign word $x^f_{i,j}$. For an alignment $a_j = \langle C_j, x^f_{i,j}\rangle$, only the sequence of characters $C_j = \{x^c_{i,j_1}, x^c_{i,j_2}, ..., x^c_{i,j_k}\}$ $\forall d \in [1, k-1], j_{d+1} - j_d = 1$ constitutes a valid candidate word. For the whole bilingual corpus, we assign each character in the candidate words with a word boundary tag $T \in \{B, M, E, S\}$, and then count across the entire corpus to collect the tag distributions $r_i = \{r_{i,t}; t \in T\}$ for each type $x^c_{i,j-1}x^c_{i,j}x^c_{i,j+1}$.

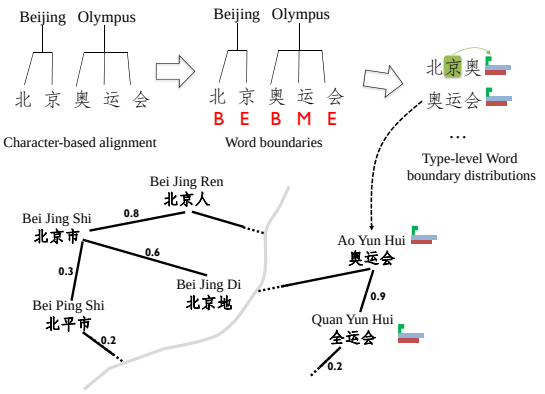


Figure 1: An example of similarity graph over character-level trigrams (types).

3.2 Constraints Encoded by Graph Propagation Expression

The previous step contributes to generate bilingual segmentation supervisions, i.e., type-level word boundary distributions. An intuitive manner is to directly leverage the induced boundary distributions as label constraints to regularize segmentation model learning, based on a constrained learning algorithm. This study, however, makes further efforts to elevate the positive effects of the bilingual knowledge via the graph propagation technique. We adopt a similarity graph to encode the learned type-level word boundary distributions. The GP expression will be defined as a PR constraint in Section 3.3 that reflects the interactions between the graph and the CRFs model. In other words, GP is integrated with estimation of parametric structural model. This is greatly different from the prior pipelined approaches (Subramanya et al., 2010; Das and Petrov, 2011; Zeng et al., 2013), where GP is run first and its propagated outcomes are then used to bias the structural model. This work seeks to capture the GP benefits during the modeling of sequential correlations.

In what follows, the graph setting and propagation expression are introduced. As in conventional GP examples (Das and Smith, 2012), a similarity graph $\mathcal{G} = (V, E)$ is constructed over N types extracted from Chinese training data, including treebank $\mathcal{D}^c_l$ and bitexts $\mathcal{D}^c_u$. Each vertex V_i has a $|T|$-dimensional estimated measure $v_i = \{v_{i,t}; t \in T\}$ representing a probability distribution on word boundary tags. The induced type-level word boundary distributions $r_i = \{r_{i,t}; t \in T\}$ are empirical measures for the corresponding M graph vertices. The edges $E \in V_i \times V_j$ connect all the vertices. Scores between pairs of graph vertices (types), w_{ij}, refer to the similarities of their syntactic environment, which are computed following the method in (Subramanya et al., 2010; Das and Petrov, 2011; Zeng et al., 2013). The similarities are measured based on co-occurrence statistics over a set of predefined features (introduced in Section 4.1). Specifically, the point-wise mutual information (PMI) values, between vertices and each feature instantiation that they have in common, are summed to sparse vectors, and their cosine distances are computed as the similarities. The nature of this similarity graph enforces that the connected types with high weights appearing in different texts should have similar word boundary distributions.

The quality (smoothness) of the similarity graph can be estimated by using a standard propagation function, as shown in Equation 1. The square-loss criterion (Zhu et al., 2003; Bengio et al., 2006) is used to formulate this function:

$$\mathcal{P}(v) = \sum_{t=1}^{T}\Bigg(\sum_{i=1}^{M}(v_{i,t} - r_{i,t})^2 + \mu\sum_{j=1}^{N}\sum_{i=1}^{N} w_{ij}(v_{i,t} - v_{j,t})^2 + \rho\sum_{i=1}^{N}(v_{i,t})^2\Bigg) \quad (1)$$

The first term in this equation refers to seed matches that compute the distances between the estimated measure v_i and the empirical probabilities r_i. The second term refers to edge smoothness that measures how vertices v_i are smoothed with respect to the graph. Two types connected by an edge with high weight should be assigned similar word boundary distributions. The third term, a ℓ_2 norm, evaluates the distribution sparsity (Das and

Smith, 2012) per vertex. Typically, the GP process amounts to an optimization process with respect to parameter v such that Equation 1 is minimized. This propagation function can be used to reflect the graph smoothness, where the higher the score, the lower the smoothness.

3.3 PR Learning with GP Constraint

Our learning problem belongs to semi-supervised learning (SSL), as the training is done on treebank labeled data $(\mathrm{X}_L, \mathrm{Y}_L) = \{(\mathrm{x}_1, \mathrm{y}_1), ..., (\mathrm{x}_l, \mathrm{y}_l)\}$, and bilingual unlabeled data $(\mathrm{X}_U) = \{\mathrm{x}_1, ..., \mathrm{x}_u\}$ where $\mathrm{x}_i = \{x^1, ..., x^m\}$ is an input word sequence and $\mathrm{y}_i = \{y^1, ..., y^m\}, y \in T$ is its corresponding label sequence. Supervised linear-chain CRFs can be modeled in a standard conditional log-likelihood objective with a Gaussian prior:

$$\mathcal{L}(\theta) = p_\theta(\mathrm{y}_i|\mathrm{x}_i) - \frac{\|\theta\|^2}{2\sigma} \tag{2}$$

The conditional probabilities p_θ are expressed as a log-linear form:

$$p_\theta(\mathrm{y}_i|\mathrm{x}_i) = \frac{\exp(\sum_{k=1}^{m} \theta^{\mathrm{T}} f(y_i^{k-1}, y_i^k, \mathrm{x}_i))}{Z_\theta(\mathrm{x}_i)} \tag{3}$$

Where $Z_\theta(\mathrm{x}_i)$ is a partition function that normalizes the exponential form to be a probability distribution, and $f(y_i^{k-1}, y_i^k, \mathrm{x}_i)$ are arbitrary feature functions.

In our setting, the CRFs model is required to learn from unlabeled data. This work employs the posterior regularization (PR) framework[3] (Ganchev et al., 2010) to bias the CRFs model's learning on unlabeled data, under a constraint encoded by the graph propagation expression. It is expected that similar types in the graph should have approximated expected taggings under the CRFs model. We follow the approach introduced by (He et al., 2013) to set up a penalty-based PR objective with GP: the CRFs likelihood is modified by adding a regularization term, as shown in Equation 4, representing the constraints:

$$\mathcal{R}_U(\theta, q) = \mathrm{KL}(q||p_\theta) + \lambda \mathcal{P}(v) \tag{4}$$

Rather than regularize CRFs model's posteriors $p_\theta(\mathcal{Y}|\mathrm{x}_i)$ directly, our model uses an auxiliary distribution $q(\mathcal{Y}|\mathrm{x}_i)$ over the possible labelings $\mathcal{Y}$ for x_i, and penalizes the CRFs marginal log-likelihood by a **KL-divergence** term[4], representing the distance between the estimated posteriors p and the desired posteriors q, as well as a **penalty** term, formed by the GP function. The hyperparameter λ is used to control the impacts of the penalty term. Note that the penalty is fired if the graph score computed based on the expected taggings given by the current CRFs model is increased vis-a-vis the previous training iteration. This nature requires that the penalty term $\mathcal{P}(v)$ should be formed as a function of posteriors q over CRFs model predictions[5], i.e., $\mathcal{P}(q)$. To state this, a mapping $\mathcal{M} : (\{1, ..., u\}, \{1, ..., m\}) \rightarrow V$ from words in the corpus to vertices in the graph is defined. We can thus decompose $v_{i,t}$ into a function of q as follows:

$$v_{i,t} = \frac{\sum_{a=1}^{u} \sum_{\substack{b=1; \\ \mathcal{M}(a,b)=V_i}}^{m} \sum_{c=1}^{T} \sum_{y \in \mathcal{Y}} \mathbf{1}(y^b = t, y^{b-1} = c) q(\mathrm{y}|\mathrm{x}_a)}{\sum_{a=1}^{u} \sum_{b=1}^{m} \mathbf{1}(\mathcal{M}(a,b) = V_i)} \tag{5}$$

The final learning objective combines the CRFs likelihood with the PR regularization term: $\mathcal{J}(\theta, q) = \mathcal{L}(\theta) + \mathcal{R}_U(\theta, q)$. This joint objective, over θ and q, can be optimized by an expectation maximization (EM) style algorithm as reported in (Ganchev et al., 2010). We start from initial parameters θ^0, estimated by supervised CRFs model training on treebank data. The E-step is to minimize $\mathcal{R}_U(\theta, q)$ over the posteriors q that are constrained to the probability simplex. Since the penalty term $\mathcal{P}(v)$ is a non-linear form, the optimization method in (Ganchev et al., 2010) via projected gradient descent on the dual is inefficient[6]. This study follows the optimization method (He et al., 2013) that uses exponentiated gradient descent (EGD) algorithm. It allows that the variable update expression, as shown in Equation 6, takes a multiplicative rather than an additive form.

$$q^{(w+1)}(\mathrm{y}|\mathrm{x}_i) = q^{(w)}(\mathrm{y}|\mathrm{x}_i) \exp(-\eta \frac{\partial \mathcal{R}}{\partial q^{(w)}(\mathrm{y}|\mathrm{x}_i)}) \tag{6}$$

where the parameter η controls the optimization rate in the E-step. With the contributions from

[3]The readers are refered to the original paper of Ganchev et al. (2010).

[4]The form of KL term: $\mathrm{KL}(q||p) = \sum_{q \in \mathcal{Y}} q(\mathrm{y}) \log \frac{q(\mathrm{y})}{p(\mathrm{y})}$.

[5]The original PR setting also requires that the penalty term should be a linear (Ganchev et al., 2010) or non-linear (He et al., 2013) function on q.

[6]According to (He et al., 2013), the dual of quadratic program implies an expensive matrix inverse.

the E-step that further encourage q and p to agree, the M-step aims to optimize the objective $\mathcal{J}(\theta, q)$ with respect to θ. The M-step is similar to the standard CRFs parameter estimation, where the gradient ascent approach still works. This EM-style approach monotonically increases $\mathcal{J}(\theta, q)$ and thus is guaranteed to converge to a local optimum.

$$\begin{aligned}\textbf{E-step:}\quad & q^{(t+1)} = \arg\min_{q} \mathcal{R}_U(\theta^{(t)}, q^{(t)}) \\ \textbf{M-step:}\quad & \theta^{(t+1)} = \arg\max_{\theta} \mathcal{L}(\theta) \\ & + \delta \sum_{i=1}^{u} \sum_{\mathrm{y} \in \mathcal{Y}} q^{(t+1)}(\mathrm{y}|\mathrm{x}_i) \log p_\theta(\mathrm{y}|\mathrm{x}_i)\end{aligned} \tag{7}$$

4 Experiments

4.1 Data and Setup

The experiments in this study evaluated the performances of various CWS models in a Chinese-to-English translation task. The influence of the word segmentation on the final translation is our main investigation. We adopted three state-of-the-art metrics, BLEU (Papineni et al., 2002), NIST (Doddington et al., 2000) and METEOR (Banerjee and Lavie, 2005), to evaluate the translation quality.

The monolingual segmented data, $\mathrm{train}_{\mathrm{TB}}$, is extracted from the Penn Chinese Treebank (CTB-7) (Xue et al., 2005), containing 51,447 sentences. The bilingual training data, $\mathrm{train}_{\mathrm{MT}}$, is formed by a large in-house Chinese-English parallel corpus (Tian et al., 2014). There are in total 2,244,319 Chinese-English sentence pairs crawled from online resources, concentrated in 5 different domains including *laws*, *novels*, *spoken*, *news* and *miscellaneous*[7]. This in-house bilingual corpus is the MT training data as well. The target-side language model is built on over 35 million monolingual English sentences, $\mathrm{train}_{\mathrm{LM}}$, crawled from online resources. The NIST evaluation campaign data, MT-03 and MT-05, are selected to comprise the MT development data, $\mathrm{dev}_{\mathrm{MT}}$, and testing data, $\mathrm{test}_{\mathrm{MT}}$, respectively.

For the settings of our model, we adopted the standard feature templates introduced by Zhao et al. (2006) for CRFs. The character-based alignment for achieving the "chars-to-word" mappings is accomplished by GIZA++ aligner (Och and Ney, 2003). For the GP, a 10-NNs similarity graph was constructed[8]. Following (Subramanya et al., 2010; Zeng et al., 2013), the features used to compute similarities between vertices were (Suppose given a type "$w_2w_3w_4$" surrounding contexts "$w_1w_2w_3w_4w_5$"): unigram (w_3), bigram (w_1w_2, w_4w_5, w_2w_4), trigram ($w_2w_3w_4$, $w_2w_4w_5$, $w_1w_2w_4$), trigram+context ($w_1w_2w_3w_4w_5$) and character classes in number, punctuation, alphabetic letter and other ($t(w_2)t(w_3)t(w_4)$). There are four hyperparameters in our model to be tuned by using the development data ($\mathrm{dev}_{\mathrm{MT}}$) among the following settings: for the graph propagation, $\mu \in \{0.2, 0.5, 0.8\}$ and $\rho \in \{0.1, 0.3, 0.5, 0.8\}$; for the PR learning, $\lambda \in \{0 \leq \lambda_i \leq 1\}$ and $\sigma \in \{0 \leq \sigma_i \leq 1\}$ where the step is 0.1. The best performed joint settings, $\mu = 0.5, \rho = 0.5, \lambda = 0.9$ and $\sigma = 0.8$, were used to measure the final performance.

The MT experiment was conducted based on a standard log-linear phrase-based SMT model. The GIZA++ aligner was also adopted to obtain word alignments (Och and Ney, 2003) over the segmented bitexts. The heuristic strategy of *grow-diag-final-and* (Koehn et al., 2007) was used to combine the bidirectional alignments for extracting phrase translations and reordering tables. A 5-gram language model with Kneser-Ney smoothing was trained with SRILM (Stolcke, 2002) on monolingual English data. Moses (Koehn et al., 2007) was used as decoder. The Minimum Error Rate Training (MERT) (Och, 2003) was used to tune the feature parameters on development data.

4.2 Various Segmentation Models

To provide a thorough analysis, the MT experiments in this study evaluated three baseline segmentation models and two off-the-shelf models, in addition to four variant models that also employ the bilingual constraints. We start from three baseline models:

- **Character Segmenter (CS)**: this model simply divides Chinese sentences into sequences of characters.
- **Supervised Monolingual Segmenter (SMS)**: this model is trained by CRFs on treebank training data ($\mathrm{train}_{\mathrm{TB}}$). The same feature templates (Zhao et al., 2006) are used. The standard four-tags (**B**, **M**, **E** and **S**) were used

[7]The in-house corpus has been manually validated, in a long process that exceeded 500 hours.

[8]We evaluated graphs with top k (from 3 to 20) nearest neighbors on development data, and found that the performance converged beyond 10-NNs.

as the labels. The stochastic gradient descent is adopted to optimize the parameters.

- **Unsupervised Bilingual Segmenter (UBS)**: this model is trained on the bitexts (trainMT) following the approach introduced in (Ma and Way, 2009). The optimal set of the model parameter values was found on $\mathrm{dev_{MT}}$ to be $k = 3, t_{AC} = 0.0$ and $t_{COOC} = 15$.

The comparison candidates also involve two popular off-the-shelf segmentation models:

- **Stanford Segmenter**: this model, trained by Chang et al. (2008), treats CWS as a binary word boundary decision task. It covers several features specific to the MT task, e.g., external lexicons and proper noun features.

- **ICTCLAS Segmenter**: this model, trained by Zhang et al. (2003), is a hierarchical HMM segmenter that incorporates parts-of-speech (POS) information into the probability models and generates multiple HMM models for solving segmentation ambiguities.

This work also evaluated four variant models[9] that perform alternative ways to incorporate the bilingual constraints based on two state-of-the-art graph-based SSL approaches.

- **Self-training Segmenters (STS)**: two variant models were defined by the approach reported in (Subramanya et al., 2010) that uses the supervised CRFs model's decodings, incorporating empirical and constraint information, for unlabeled examples as additional labeled data to retrain a CRFs model. One variant (STS-NO-GP) skips the GP step, directly decoding with type-level word boundary probabilities induced from bitexts, while the other (STS-GP-PL) runs the GP at first and then decodes with GP outcomes. The optimal hyperparameter values were found to be: STS-NO-GP ($\alpha = 0.8$) and $\eta = 0.6$) and STS-GP-PL ($\mu = 0.5, \rho = 0.3, \alpha = 0.8$ and $\eta = 0.6$).

- **Virtual Evidences Segmenters (VES)**: Two variant models based on the approach in (Zeng et al., 2013) were defined. The type-level word boundary distributions, induced by the character-based alignment (VES-NO-GP), and the graph propagation (VES-GP-PL), are regarded as virtual evidences to bias CRFs model's learning on the unlabeled data. The optimal hyperparameter values were found to be: VES-NO-GP ($\alpha = 0.7$) and VES-GP-PL ($\mu = 0.5, \rho = 0.3$ and $\alpha = 0.7$).

[9]Note that there are two variant models working with GP. To be fair, the same similarity graph settings introduced in this paper were used.

4.3 Main Results

Table 1 summarizes the final MT performance on the MT-05 test data, evaluated with ten different CWS models. In what follows, we summarized four major observations from the results. Firstly, as expected, having word segmentation does help Chinese-to-English MT. All other nine CWS models outperforms the CS baseline which does not try to identify Chinese words at all. Secondly, the other two baselines, SMS and UBS, are on a par with each other, showing less than 0.36 average performance differences on the three evaluation metrics. This outcome validated that the models, trained by either the treebank or the bilingual data, performed reasonably well. But they only capture partial segmentation features so that less gains for SMT are achieved when comparing to other sophisticated models. Thirdly, we notice that the two off-the-shelf models, Stanford and ICTCLAS, just brought minor improvements over the SMS baseline, although they are trained using richer supervisions. This behaviour illustrates that the conventional optimizations to the monolingual supervised model, e.g., accumulating more supervised data or predefined segmentation properties, are insufficient to help model for achieving better segmentations for SMT. Finally, highlighting the five models working with the bilingual constraints, most of them can achieve significant gains over the other ones without using the bilingual constraints. This strongly demonstrates that bilingually-learned segmentation knowledge does helps CWS for SMT. The models working with G-P, STS-GP-PL, VES-GP-PL and ours outperform all others. We attribute this to the role of GP in assisting the spread of bilingual knowledge on the Chinese side. Importantly, it can be observed that our model outperforms STS-GP, VES-GP, which greatly supports that joint learning of CRFs and GP can alleviate the error transfer by the pipelined models. This is one of the most crucial findings in this study. Overall, the boldface numbers in the last row illustrate that our model obtains average improvements of 1.89, 1.76 and 1.61 on BLEU,

NIST and METEOR over others.

Models	BLEU	NIST	METEOR
CS	29.38	59.85	54.07
SMS	30.05	61.33	55.95
UBS	30.15	61.56	55.39
Stanford	30.40	61.94	56.01
ICTCLAS	30.29	61.26	55.72
STS-NO-GP	31.47	62.35	56.12
STS-GP-PL	31.94	63.20	57.09
VES-NO-GP	31.98	62.63	56.59
VES-GP-PL	32.04	63.49	57.34
Our Model	**32.75**	**63.72**	**57.64**

Table 1: Translation performances (%) on MT-05 testing data by using ten different CWS models.

4.4 Analysis & Discussion

This section aims to further analyze the three primary observations concluded in Section 4.3: i) word segmentation is useful to SMT; ii) the treebank and the bilingual segmentation knowledge are helpful, performing segmentation of different nature; and iii) the bilingual constraints lead to learn segmentations better tailored for SMT.

The first observation derives from the comparisons between the CS baseline and other models. Our results, showing the significant CWS benefits to SMT, are consistent with the works reported in the literature (Xu et al., 2004; Chang et al., 2008). In our experiment, two additional evidences found in the translation model are provided to further support that NO tokenization of Chinese (i.e., the CS model's output) could harm the MT system. First, the SMT phrase extraction, i.e., building "phrases" on top of the character sequences, cannot fully capture all meaningful segmentations produced by the CS model. The character based model leads to missing some useful longer phrases, and to generate many meaningless or redundant translations in the phrase table. Moreover, it is affected by translation ambiguities, caused by the cases where a Chinese character has very different meanings in different contextual environments.

The second observation shifts the emphasis to SMS and UBS, based on the treebank and the bilingual segmentation, respectively. Our results show that both segmentation patterns can bring positive effects to MT. Through analyzing both models' segmentations for $\mathrm{train}_{\mathrm{MT}}$ and $\mathrm{test}_{\mathrm{MT}}$, we attempted to get a closer inspection on the segmentation preferences and their influence on MT. Our first finding is that the segmentation consensuses between SMS and UBS are positive to MT. There have about 35% identical segmentations produced by the two models. If these identical segmentations are removed, and the experiments are rerun, the translation scores decrease (on average) by 0.50, 0.85 and 0.70 on BLEU, NIST and METEOR, respectively. Our second finding is that SMS exhibits better segmentation consistency than UBS. One representative example is the segmentations for "孤零零 (lonely)". All the outputs of SMS were "孤零零", while UBS generated three ambiguous segmentations, "孤(alone)_零零(double zero)", "孤 零(lonely)_零(zero)" and "孤(alone)_零(zero)_零(zero)". The segmentation consistency of SMS rests on the high-quality treebank data and the robust CRFs tagging model. On the other hand, the advantage of UBS is to capture the segmentations matching the aligned target words. For example, UBS grouped "国(country)_际(border)_间(between)" to a word "国 际 间(international)", rather than two words "国际(international)_间(between)" (as given by SMS), since these three characters are aligned to a single English word "international". The above analysis shows that SMS and UBS have their own merits and combining the knowledge derived from both segmentations is highly encouraged.

The third observation concerns the great impact of the bilingual constraints to the segmentation models in the MT task. The use of the bilingual constraints is the prime objective of this study. Our first contribution for this purpose is on using the word boundary distributions to capture the bilingual segmentation supervisions. This representation contributes to reduce the negative impacts of erroneous "chars-to-word" alignments. The ambiguous types (having relatively uniform boundary distribution), caused by alignment errors, cannot directly bias the model tagging preferences. Furthermore, the word boundary distributions are convenient to make up the learning constraints over the labelings among various constrained learning approaches. They have successfully played in three types of constraints for our experiments: PR penalty (Our model), decoding constraints in self-training (STS) and virtual evidences (VES). The second contribution is the use of GP, illustrated by STS-GP-PL, VES-GP-PL and

Our model. The major effect is to multiply the impacts of the bilingual knowledge through the similarity graph. The graph vertices (types)[10], without any supervisions, can learn the word boundary information from their similar types (neighborhoods) having the empirical boundary probabilities. The segmentations given by the three GP models show about 70% positive segmentation changes, affected by the unlabeled graph vertices, with respect to the ones given by the NO-GP models, STS-NO-GP and VES-NO-GP. In our opinion, the learning mechanism of our approach, joint coupling of GP and CRFs, rather than the pipelined one as the other two models, contributes to maximizing the graph smoothness effects to the CRFs estimation so that the error propagation of the pipelined approaches is alleviated.

5 Conclusion

This paper proposed a novel CWS model for the SMT task. This model aims to maintain the linguistic segmentation supervisions from treebank data and simultaneously integrate useful bilingual segmentations induced from the bitexts. This objective is accomplished by three main steps: 1) learn word boundaries from character-based alignments; 2) encode the learned word boundaries into a GP constraint; and 3) training a CRFs model, under the GP constraint, by using the PR framework. The empirical results indicate that the proposed model can yield better segmentations for SMT.

Acknowledgments

The authors are grateful to the Science and Technology Development Fund of Macau and the Research Committee of the University of Macau (Grant No. MYRG076 (Y1-L2)-FST13-WF and MYRG070 (Y1-L2)-FST12-CS) for the funding support for our research. The work of Isabel Trancoso was supported by national funds through FCT-Fundação para a Ciência e a Tecnologia, under project PEst-OE/EEI/LA0021/2013. The authors also wish to thank the anonymous reviewers for many helpful comments.

[10]This experiment yielded a similarity graph that consists of 11,909,620 types from $\mathrm{train}_{\mathrm{TB}}$ and $\mathrm{train}_{\mathrm{MT}}$, where there have 8,593,220 (72.15%) types without any empirical boundary distributions.

References

Yasemin Altun, David McAllester, and Mikhail Belkin. 2006. Maximum margin semi-supervised learning for structured variables. *Advances in Neural Information Processing Systems*, 18:33.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72. Association for Computational Linguistics.

Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. 2006. Label propagation and quadratic criterion. *Semi-Supervised Learning*, pages 193–216.

Pi-Chuan Chang, Michel Galley, and Christopher D Manning. 2008. Optimizing Chinese word segmentation for machine translation performance. In *Proceedings of WMT*, pages 224–232. Association for Computational Linguistics.

Tagyoung Chung and Daniel Gildea. 2009. Unsupervised tokenization for machine translation. In *Proceedings of EMNLP*, pages 718–726. Association for Computational Linguistics.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of ACL*, pages 600–609. Association for Computational Linguistics.

Dipanjan Das and Noah A Smith. 2012. Graph-based lexicon expansion with sparsity-inducing penalties. In *Proceedings of NAACL*, pages 677–687. Association for Computational Linguistics.

George R. Doddington, Mark A. Przybocki, Alvin F. Martin, and Douglas A. Reynolds. 2000. The nist speaker recognition evaluation–overview, methodology, systems, results, perspective. *Speech Communication*, 31(2):225–254.

Kuzman Ganchev, Jõao Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *The Journal of Machine Learning Research*, 11:2001–2049.

Luheng He, Jennifer Gillenwater, and Ben Taskar. 2013. Graph-based posterior regularization for semi-supervised structured prediction. In *Proceedings of CoNLL*, page 38. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL on Interactive Poster and Demonstration Sessions*, pages 177–180. Association for Computational Linguistics.

Yanjun Ma and Andy Way. 2009. Bilingually motivated domain-adapted word segmentation for statistical machine translation. In *Proceedings of EACL*, pages 549–557. Association for Computational Linguistics.

Gideon S. Mann and Andrew McCallum. 2008. Generalized expectation criteria for semi-supervised learning of conditional random fields. In *Proceedings of ACL*, pages 870–878. Association for Computational Linguistics.

Andrew McCallum, Gideon Mann, and Gregory Druck. 2007. Generalized expectation criteria. *Computer Science Technical Note, University of Massachusetts, Amherst, MA*.

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

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*, pages 160–167. Association for Computational Linguistics.

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

Michael Paul, Finch Andrew, and Sumita Eiichiro. 2011. Integration of multiple bilingually-trained segmentation schemes into statistical machine translation. *IEICE Transactions on Information and Systems*, 94(3):690–697.

Andreas Stolcke. 2002. SRILM-an extensible language modeling toolkit. In *Proceedings of Interspeech*.

Amarnag Subramanya, Slav Petrov, and Fernando Pereira. 2010. Efficient graph-based semi-supervised learning of structured tagging models. In *Proceedings of EMNLP*, pages 167–176. Association for Computational Linguistics.

Liang Tian, Derek F. Wong, Lidia S. Chao, Paulo Quaresma, Francisco Oliveira, Shuo Li, Yiming Wang, and Yi Lu. 2014. UM-Corpus: A large English-Chinese parallel corpus for statistical machine translation. In *Proceedings of LREC*. European Language Resources Association.

Ning Xi, Guangchao Tang, Xinyu Dai, Shujian Huang, and Jiajun Chen. 2012. Enhancing statistical machine translation with character alignment. In *Proceedings of ACL*, pages 285–290. Association for Computational Linguistics.

Jia Xu, Richard Zens, and Hermann Ney. 2004. Do we need Chinese word segmentation for statistical machine translation? In *Proceedings of the Third SIGHAN Workshop on Chinese Language Learning*, pages 122–128. Association for Computational Linguistics.

Jia Xu, Evgeny Matusov, Richard Zens, and Hermann Ney. 2005. Integrated Chinese word segmentation in statistical machine translation. In *Proceedings of IWSLT*, pages 216–223. Association for Computational Linguistics.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238.

Xiaodong Zeng, Derek F. Wong, Lidia S. Chao, and Isabel Trancoso. 2013. Graph-based semi-supervised model for joint Chinese word segmentation and part-of-speech tagging. In *Proceedings of ACL*, pages 770–779. Association for Computational Linguistics.

Xiaodong Zeng, Derek F. Wong, Lidia S. Chao, Isabel Trancoso, Liangye He, and Qiuping Huang. 2014. Lexicon expansion for latent variable grammars. *Pattern Recognition Letters*, 42:47–55.

Hua-Ping Zhang, Hong-Kui Yu, De-Yi Xiong, and Qun Liu. 2003. HHMM-based Chinese lexical analyzer ICTCLAS. In *Proceedings of the Second SIGHAN Workshop on Chinese Language Processing*, pages 184–187. Association for Computational Linguistics.

Ruiqiang Zhang, Keiji Yasuda, and Eiichiro Sumita. 2008. Improved statistical machine translation by multiple Chinese word segmentation. In *Proceedings of WMT*, pages 216–223. Association for Computational Linguistics.

Hai Zhao, Chang-Ning Huang, and Mu Li. 2006. An improved Chinese word segmentation system with conditional random field. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*. Association for Computational Linguistics.

Hai Zhao, Masao Utiyama, Eiichiro Sumita, and Bao-Liang Lu. 2013. An empirical study on word segmentation for Chinese machine translation. In *Computational Linguistics and Intelligent Text Processing*, pages 248–263. Springer.

Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of ICML*, volume 3, pages 912–919.

Ling Zhu, Derek F. Wong, and Lidia S. Chao. 2014. Unsupervised chunking based on graph propagation from bilingual corpus. *The Scientific World Journal*, 2014(401943):10.

Fast and Robust Neural Network Joint Models for Statistical Machine Translation

Jacob Devlin, Rabih Zbib, Zhongqiang Huang,
Thomas Lamar, Richard Schwartz, and **John Makhoul**
Raytheon BBN Technologies, 10 Moulton St, Cambridge, MA 02138, USA
{jdevlin,rzbib,zhuang,tlamar,schwartz,makhoul}@bbn.com

Abstract

Recent work has shown success in using neural network language models (NNLMs) as features in MT systems. Here, we present a novel formulation for a neural network *joint* model (NNJM), which augments the NNLM with a source context window. Our model is purely lexicalized and can be integrated into any MT decoder. We also present several variations of the NNJM which provide significant additive improvements.

Although the model is quite simple, it yields strong empirical results. On the NIST OpenMT12 Arabic-English condition, the NNJM features produce a gain of +3.0 BLEU on top of a powerful, feature-rich baseline which already includes a target-only NNLM. The NNJM features also produce a gain of +6.3 BLEU on top of a simpler baseline equivalent to Chiang's (2007) original Hiero implementation.

Additionally, we describe two novel techniques for overcoming the historically high cost of using NNLM-style models in MT decoding. These techniques speed up NNJM computation by a factor of 10,000x, making the model as fast as a standard back-off LM.

This work was supported by DARPA/I2O Contract No. HR0011-12-C-0014 under the BOLT program (Approved for Public Release, Distribution Unlimited). The views, opinions, and/or findings contained in this article are those of the author and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

1 Introduction

In recent years, neural network models have become increasingly popular in NLP. Initially, these models were primarily used to create n-gram neural network language models (NNLMs) for speech recognition and machine translation (Bengio et al., 2003; Schwenk, 2010). They have since been extended to translation modeling, parsing, and many other NLP tasks.

In this paper we use a basic neural network architecture and a lexicalized probability model to create a powerful MT decoding feature. Specifically, we introduce a novel formulation for a neural network joint model (NNJM), which augments an n-gram target language model with an m-word source window. Unlike previous approaches to joint modeling (Le et al., 2012), our feature can be easily integrated into any statistical machine translation (SMT) decoder, which leads to substantially larger improvements than k-best rescoring only. Additionally, we present several variations of this model which provide significant additive BLEU gains.

We also present a novel technique for training the neural network to be *self-normalized*, which avoids the costly step of posteriorizing over the entire vocabulary in decoding. When used in conjunction with a *pre-computed* hidden layer, these techniques speed up NNJM computation by a factor of 10,000x, with only a small reduction on MT accuracy.

Although our model is quite simple, we obtain strong empirical results. We show primary results on the NIST OpenMT12 Arabic-English condition. The NNJM features produce an improvement of +3.0 BLEU on top of a baseline that is already better than the 1st place MT12 result and includes

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1370–1380,
Baltimore, Maryland, USA, June 23-25 2014.

a powerful NNLM. Additionally, on top of a simpler decoder equivalent to Chiang's (2007) original Hiero implementation, our NNJM features are able to produce an improvement of +6.3 BLEU – as much as all of the other features in our strong baseline system combined.

We also show strong improvements on the NIST OpenMT12 Chinese-English task, as well as the DARPA BOLT (Broad Operational Language Translation) Arabic-English and Chinese-English conditions.

2 Neural Network Joint Model (NNJM)

Formally, our model approximates the probability of target hypothesis T conditioned on source sentence S. We follow the standard n-gram LM decomposition of the target, where each target word t_i is conditioned on the previous $n-1$ target words. To make this a *joint* model, we also condition on source context vector $\mathcal{S}_i$:

$$P(T|S) \approx \Pi_{i=1}^{|T|} P(t_i|t_{i-1}, \cdots, t_{i-n+1}, \mathcal{S}_i)$$

Intuitively, we want to define $\mathcal{S}_i$ as the window that is most relevant to t_i. To do this, we first say that each target word t_i is *affiliated* with exactly one source word at index a_i. $\mathcal{S}_i$ is then the m-word source window centered at a_i:

$$\mathcal{S}_i = s_{a_i - \frac{m-1}{2}}, \cdots, s_{a_i}, \cdots, s_{a_i + \frac{m-1}{2}}$$

This notion of *affiliation* is derived from the word alignment, but unlike word alignment, each target word must be affiliated with exactly one non-NULL source word. The affiliation heuristic is very simple:

(1) If t_i aligns to exactly one source word, a_i is the index of the word it aligns to.
(2) If t_i align to multiple source words, a_i is the index of the aligned word in the middle.[1]
(3) If t_i is unaligned, we inherit its affiliation from the closest aligned word, with preference given to the right.[2]

An example of the NNJM context model for a Chinese-English parallel sentence is given in Figure 1.

For all of our experiments we use $n = 4$ and $m = 11$. It is clear that this model is effectively an $(n+m)$-gram LM, and a 15-gram LM would be far too sparse for standard probability models such as Kneser-Ney back-off (Kneser and Ney, 1995) or Maximum Entropy (Rosenfeld, 1996). Fortunately, neural network language models are able to elegantly scale up and take advantage of arbitrarily large context sizes.

2.1 Neural Network Architecture

Our neural network architecture is almost identical to the original feed-forward NNLM architecture described in Bengio et al. (2003).

The input vector is a 14-word context vector (3 target words, 11 source words), where each word is mapped to a 192-dimensional vector using a shared mapping layer. We use two 512-dimensional hidden layers with $tanh$ activation functions. The output layer is a softmax over the entire output vocabulary.

The input vocabulary contains 16,000 source words and 16,000 target words, while the output vocabulary contains 32,000 target words. The vocabulary is selected by frequency-sorting the words in the parallel training data. Out-of-vocabulary words are mapped to their POS tag (or `OOV`, if POS is not available), and in this case $P(POS_i|t_{i-1}, \cdots)$ is used directly without further normalization. Out-of-bounds words are represented with special tokens `<src>`, `</src>`, `<trg>`, `</trg>`.

We chose these values for the hidden layer size, vocabulary size, and source window size because they seemed to work best on our data sets – larger sizes did not improve results, while smaller sizes degraded results. Empirical comparisons are given in Section 6.5.

2.2 Neural Network Training

The training procedure is identical to that of an NNLM, except that the parallel corpus is used instead of a monolingual corpus. Formally, we seek to maximize the log-likelihood of the training data:

$$L = \sum_i \log(P(x_i))$$

where x_i is the training sample, with one sample for every target word in the parallel corpus.

Optimization is performed using standard back propagation with stochastic gradient ascent (LeCun et al., 1998). Weights are randomly initialized in the range of $[-0.05, 0.05]$. We use an initial learning rate of 10^{-3} and a minibatch size of

[1] We arbitrarily round down.

[2] We have found that the affiliation heuristic is robust to small differences, such as left vs. right preference.

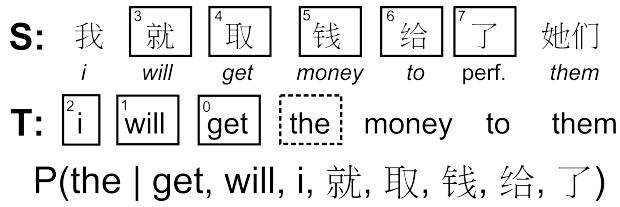


Figure 1: Context vector for target word "the", using a 3-word target history and a 5-word source window (i.e., $n = 4$ and $m = 5$). Here, "the" inherits its affiliation from "money" because this is the first aligned word to its right. The number in each box denotes the index of the word in the context vector. This indexing must be consistent across samples, but the absolute ordering does not affect results.

128.[3] At every epoch, which we define as 20,000 minibatches, the likelihood of a validation set is computed. If this likelihood is worse than the previous epoch, the learning rate is multiplied by 0.5. The training is run for 40 epochs. The training data ranges from 10-30M words, depending on the condition. We perform a basic weight update with no L2 regularization or momentum. However, we have found it beneficial to clip each weight update to the range of [-0.1, 0.1], to prevent the training from entering degenerate search spaces (Pascanu et al., 2012).

Training is performed on a single Tesla K10 GPU, with each epoch (128*20k = 2.6M samples) taking roughly 1100 seconds to run, resulting in a total training time of ∼12 hours. Decoding is performed on a CPU.

2.3 Self-Normalized Neural Network

The computational cost of NNLMs is a significant issue in decoding, and this cost is dominated by the output softmax over the entire target vocabulary. Even class-based approaches such as Le et al. (2012) require a 2-20k shortlist vocabulary, and are therefore still quite costly.

Here, our goal is to be able to use a fairly large vocabulary without word classes, and to simply avoid computing the entire output layer at decode time.[4] To do this, we present the novel technique of *self-normalization*, where the output layer scores are close to being probabilities without explicitly performing a softmax.

Formally, we define the standard softmax log likelihood as:

$$\begin{aligned} \log(P(x)) &= \log\left(\frac{e^{U_r(x)}}{Z(x)}\right) \\ &= U_r(x) - \log(Z(x)) \\ Z(x) &= \Sigma_{r'=1}^{|V|} e^{U_{r'}(x)} \end{aligned}$$

where x is the sample, U is the raw output layer scores, r is the output layer row corresponding to the observed target word, and $Z(x)$ is the softmax normalizer.

If we could guarantee that $\log(Z(x))$ were always equal to 0 (i.e., $Z(x)$ = 1) then at decode time we would only have to compute row r of the output layer instead of the whole matrix. While we cannot train a neural network with this guarantee, we can *explicitly encourage* the log-softmax normalizer to be as close to 0 as possible by augmenting our training objective function:

$$\begin{aligned} L &= \sum_i \left[\log(P(x_i)) - \alpha(\log(Z(x_i)) - 0)^2\right] \\ &= \sum_i \left[\log(P(x_i)) - \alpha \log^2(Z(x_i))\right] \end{aligned}$$

In this case, the output layer bias weights are initialized to $\log(1/|V|)$, so that the initial network is self-normalized. At decode time, we simply use $U_r(x)$ as the feature score, rather than $\log(P(x))$. For our NNJM architecture, self-normalization increases the lookup speed during decoding by a factor of ∼15x.

Table 1 shows the neural network training results with various values of the free parameter α. In all subsequent MT experiments, we use $\alpha = 10^{-1}$.

We should note that Vaswani et al. (2013) implements a method called Noise Contrastive Estimation (NCE) that is also used to train self-normalized NNLMs. Although NCE results in faster training time, it has the downside that there

[3] We do *not* divide the gradient by the minibatch size. For those who do, this is equivalent to using an initial learning rate of $10^{-3} * 128 \approx 10^{-1}$.

[4] We are not concerned with speeding up training time, as we already find GPU training time to be adequate.

Arabic BOLT Val		
α	$\log(P(x))$	$\lvert\log(Z(x))\rvert$
0	-1.82	5.02
10^{-2}	-1.81	1.35
10^{-1}	-1.83	0.68
1	-1.91	0.28

Table 1: Comparison of neural network likelihood for various α values. $\log(P(x))$ is the average log-likelihood on a held-out set. $|\log(Z(x))|$ is the mean error in log-likelihood when using $U_r(x)$ directly instead of the true softmax probability $\log(P(x))$. Note that $\alpha = 0$ is equivalent to the standard neural network objective function.

is no mechanism to control the degree of self-normalization. By contrast, our α parameter allows us to carefully choose the optimal trade-off between neural network accuracy and mean self-normalization error. In future work, we will thoroughly compare self-normalization vs. NCE.

2.4 Pre-Computing the Hidden Layer

Although self-normalization significantly improves the speed of NNJM lookups, the model is still several orders of magnitude slower than a back-off LM. Here, we present a "trick" for pre-computing the first hidden layer, which further increases the speed of NNJM lookups by a factor of 1,000x.

Note that this technique only results in a significant speedup for self-normalized, feed-forward, NNLM-style networks with *one* hidden layer. We demonstrate in Section 6.6 that using one hidden layer instead of two has minimal effect on BLEU.

For the neural network described in Section 2.1, computing the first hidden layer requires multiplying a 2689-dimensional input vector[5] with a 2689×512 dimensional hidden layer matrix. However, note that there are only 3 possible positions for each target word, and 11 for each source word. Therefore, for every word in the vocabulary, and for each position, we can pre-compute the dot product between the word embedding and the first hidden layer. These are computed offline and stored in a lookup table, which is <500MB in size.

Computing the first hidden layer now only requires 15 scalar additions for each of the 512 hidden rows – one for each word in the input vector, plus the bias. This can be reduced to just 5 scalar additions by pre-summing each 11-word source window when starting a test sentence. If our neural network has only one hidden layer and is self-normalized, the only remaining computation is 512 calls to $tanh()$ and a single 513-dimensional dot product for the final output score.[6] Thus, only ∼3500 arithmetic operations are required per n-gram lookup, compared to ∼2.8M for self-normalized NNJM without pre-computation, and ∼35M for the standard NNJM.[7]

Neural Network Speed		
Condition	lookups/sec	sec/word
Standard	110	10.9
+ Self-Norm	1500	0.8
+ Pre-Computation	1,430,000	0.0008

Table 2: Speed of the neural network computation on a single CPU thread. "lookups/sec" is the number of unique n-gram probabilities that can be computed per second. "sec/word" is the amortized cost of unique NNJM lookups in decoding, per source word.

Table 2 shows the speed of self-normalization and pre-computation for the NNJM. The decoding cost is based on a measurement of ∼1200 unique NNJM lookups per source word for our Arabic-English system.[8]

By combining self-normalization and pre-computation, we can achieve a speed of 1.4M lookups/second, which is on par with fast back-off LM implementations (Tanaka et al., 2013). We demonstrate in Section 6.6 that using the self-normalized/pre-computed NNJM results in only a very small BLEU degradation compared to the standard NNJM.

3 Decoding with the NNJM

Because our NNJM is fundamentally an n-gram NNLM with additional source context, it can easily be integrated into any SMT decoder. In this section, we describe the considerations that must be taken when integrating the NNJM into a hierarchical decoder.

[5] 2689 = 14 words × 192 dimensions + 1 bias

[6] $tanh()$ is implemented using a lookup table.

[7] $3500 \approx 5 \times 512 + 2 \times 513$; $2.8\text{M} \approx 2 \times 2689 \times 512 + 2 \times 513$; $35\text{M} \approx 2 \times 2689 \times 512 + 2 \times 513 \times 32000$. For the sake of a fair comparison, these all use one hidden layer. A second hidden layer adds 0.5M floating point operations.

[8] This does not include the cost of *duplicate* lookups within the same test sentence, which are cached.

3.1 Hierarchical Parsing

When performing hierarchical decoding with an n-gram LM, the leftmost and rightmost $n-1$ words from each constituent must be stored in the state space. Here, we extend the state space to also include the index of the affiliated source word for these edge words. This does not noticeably increase the search space. We also train a separate lower-order n-gram model, which is necessary to compute estimate scores during hierarchical decoding.

3.2 Affiliation Heuristic

For aligned target words, the normal affiliation heuristic can be used, since the word alignment is available within the rule. For unaligned words, the normal heuristic can also be used, *except* when the word is on the edge of a rule, because then the target neighbor words are not necessarily known.

In this case, we infer the affiliation from the rule structure. Specifically, if unaligned target word t is on the right edge of an arc that covers source span $[s_i, s_j]$, we simply say that t is affiliated with source word s_j. If t is on the left edge of the arc, we say it is affiliated with s_i.

4 Model Variations

Recall that our NNJM feature can be described with the following probability:

$$\Pi_{i=1}^{|T|} P(t_i | t_{i-1}, t_{i-2}, \cdots, s_{a_i}, s_{a_i-1}, s_{a_i+1}, \cdots)$$

This formulation lends itself to several natural variations. In particular, we can reverse the translation direction of the languages, as well as the direction of the language model.

We denote our original formulation as a source-to-target, left-to-right model (**S2T/L2R**). We can train three variations using target-to-source (T2S) and right-to-left (R2L) models:

S2T/R2L

$$\Pi_{i=1}^{|T|} P(t_i | t_{i+1}, t_{i+2}, \cdots, s_{a_i}, s_{a_i-1}, s_{a_i+1}, \cdots)$$

T2S/L2R

$$\Pi_{i=1}^{|S|} P(s_i | s_{i-1}, s_{i-2}, \cdots, t_{a'_i}, t_{a'_i-1}, t_{a'_i+1}, \cdots)$$

T2S/R2L

$$\Pi_{i=1}^{|S|} P(s_i | s_{i+1}, s_{i+2}, \cdots, t_{a'_i}, t_{a'_i-1}, t_{a'_i+1}, \cdots)$$

where a'_i is the target-to-source affiliation, defined analogously to a_i.

The T2S variations cannot be used in decoding due to the large target context required, and are thus only used in k-best rescoring. The S2T/R2L variant could be used in decoding, but we have not found this beneficial, so we only use it in rescoring.

4.1 Neural Network Lexical Translation Model (NNLTM)

One issue with the S2T NNJM is that the probability is computed over every *target* word, so it does not explicitly model NULL-aligned source words. In order to assign a probability to every source word during decoding, we also train a neural network lexical translation model (NNLMT).

Here, the input context is the 11-word source window centered at s_i, and the output is the target token t_{s_i} which s_i aligns to. The probability is computed over every *source* word in the input sentence. We treat NULL as a normal target word, and if a source word aligns to multiple target words, it is treated as a single concatenated token. Formally, the probability model is:

$$\Pi_{i=1}^{|S|} P(t_{s_i} | s_i, s_{i-1}, s_{i+1}, \cdots)$$

This model is trained and evaluated like our NNJM. It is easy and computationally inexpensive to use this model in decoding, since only one neural network computation must be made for each source word.

In rescoring, we also use a T2S NNLTM model computed over every target word:

$$\Pi_{i=1}^{|T|} P(s_{t_i} | t_i, t_{i-1}, t_{i+1}, \cdots)$$

5 MT System

In this section, we describe the MT system used in our experiments.

5.1 MT Decoder

We use a state-of-the-art string-to-dependency hierarchical decoder (Shen et al., 2010). Our baseline decoder contains a large and powerful set of features, which include:

- Forward and backward rule probabilities
- 4-gram Kneser-Ney LM
- Dependency LM (Shen et al., 2010)
- Contextual lexical smoothing (Devlin, 2009)
- Length distribution (Shen et al., 2010)
- Trait features (Devlin and Matsoukas, 2012)
- Factored source syntax (Huang et al., 2013)
- 7 sparse feature types, totaling 50k features (Chiang et al., 2009)
- LM adaptation (Snover et al., 2008)

We also perform 1000-best rescoring with the following features:

- 5-gram Kneser-Ney LM
- Recurrent neural network language model (RNNLM) (Mikolov et al., 2010)

Although we consider the RNNLM to be part of our baseline, we give it special treatment in the results section because we would expect it to have the highest overlap with our NNJM.

5.2 Training and Optimization

For Arabic word tokenization, we use the MADA-ARZ tokenizer (Habash et al., 2013) for the BOLT condition, and the Sakhr[9] tokenizer for the NIST condition. For Chinese tokenization, we use a simple longest-match-first lexicon-based approach.

For word alignment, we align all of the training data with both GIZA++ (Och and Ney, 2003) and NILE (Riesa et al., 2011), and concatenate the corpora together for rule extraction.

For MT feature weight optimization, we use iterative k-best optimization with an Expected-BLEU objective function (Rosti et al., 2010).

6 Experimental Results

We present MT primary results on Arabic-English and Chinese-English for the NIST OpenMT12 and DARPA BOLT conditions. We also present a set of auxiliary results in order to further analyze our features.

6.1 NIST OpenMT12 Results

Our NIST system is fully compatible with the OpenMT12 constrained track, which consists of 10M words of high-quality parallel training for Arabic, and 25M words for Chinese.[10] The Kneser-Ney LM is trained on 5B words of data from English GigaWord. For test, we use the "Arabic-To-English Original Progress Test" (1378 segments) and "Chinese-to-English Original Progress Test + OpenMT12 Current Test" (2190 segments), which consists of a mix of newswire and web data.[11] All test segments have 4 references. Our tuning set contains 5000 segments, and is a mix of the MT02-05 eval set as well as held-out parallel training.

[9] http://www.sakhr.com

[10] We also make weak use of 30M-100M words of UN data + ISI comparable corpora, but this data provides almost no benefit.

[11] http://www.nist.gov/itl/iad/mig/openmt12results.cfm

NIST MT12 Test		
	Ar-En	**Ch-En**
	BLEU	BLEU
OpenMT12 - 1st Place	49.5	32.6
OpenMT12 - 2nd Place	47.5	32.2
OpenMT12 - 3rd Place	47.4	30.8
...	...	...
OpenMT12 - 9th Place	44.0	27.0
OpenMT12 - 10th Place	41.2	25.7
Baseline (w/o RNNLM)	48.9	33.0
Baseline (w/ RNNLM)	49.8	33.4
+ S2T/L2R NNJM (Dec)	51.2	34.2
+ S2T NNLTM (Dec)	52.0	34.2
+ T2S NNLTM (Resc)	51.9	34.2
+ S2T/R2L NNJM (Resc)	52.2	34.3
+ T2S/L2R NNJM (Resc)	52.3	34.5
+ T2S/R2L NNJM (Resc)	52.8	34.7
"Simple Hier." Baseline	43.4	30.1
+ S2T/L2R NNJM (Dec)	47.2	31.5
+ S2T NNLTM (Dec)	48.5	31.8
+ Other NNJMs (Resc)	49.7	32.2

Table 3: Primary results on Arabic-English and Chinese-English NIST MT12 Test Set. The first section corresponds to the top and bottom ranked systems from the evaluation, and are taken from the NIST website. The second section corresponds to results on top of our strongest baseline. The third section corresponds to results on top of a simpler baseline. Within each section, each row includes all of the features from previous rows. BLEU scores are mixed-case.

Results are shown in the second section of Table 3. On Arabic-English, the primary S2T/L2R NNJM gains +1.4 BLEU on top of our baseline, while the S2T NNLTM gains another +0.8, and the directional variations gain +0.8 BLEU more. This leads to a total improvement of +3.0 BLEU from the NNJM and its variations. Considering that our baseline is already +0.3 BLEU better than the 1st place result of MT12 and contains a strong RNNLM, we consider this to be quite an extraordinary improvement.[12]

For the Chinese-English condition, there is an improvement of +0.8 BLEU from the primary NNJM and +1.3 BLEU overall. Here, the baseline system is already +0.8 BLEU better than the

[12] Note that the official 1st place OpenMT12 result was our own system, so we can assure that these comparisons are accurate.

best MT12 system. The smaller improvement on Chinese-English compared to Arabic-English is consistent with the behavior of our baseline features, as we show in the next section.

6.2 "Simple Hierarchical" NIST Results

The baseline used in the last section is a highly-engineered research system, which uses a wide array of features that were refined over a number of years, and some of which require linguistic resources. Because of this, the baseline BLEU scores are much higher than a typical MT system – especially a real-time, production engine which must support many language pairs.

Therefore, we also present results using a simpler version of our decoder which emulates Chiang's original Hiero implementation (Chiang, 2007). Specifically, this means that we don't use dependency-based rule extraction, and our decoder only contains the following MT features: (1) rule probabilities, (2) n-gram Kneser-Ney LM, (3) lexical smoothing, (4) target word count, (5) concat rule penalty.

Results are shown in the third section of Table 3. The "Simple Hierarchical" Arabic-English system is -6.4 BLEU worse than our strong baseline, and would have ranked 10th place out of 11 systems in the evaluation. When the NNJM features are added to this system, we see an improvement of +6.3 BLEU, which would have ranked 1st place in the evaluation.

Effectively, this means that for Arabic-English, the NNJM features are equivalent to the combined improvements from the string-to-dependency model plus all of the features listed in Section 5.1.

For Chinese-English, the "Simple Hierarchical" system only degrades by -3.2 BLEU compared to our strongest baseline, and the NNJM features produce a gain of +2.1 BLEU on top of that.

6.3 BOLT Web Forum Results

DARPA BOLT is a major research project with the goal of improving translation of informal, dialectical Arabic and Chinese into English. The BOLT domain presented here is "web forum," which was crawled from various Chinese and Egyptian Internet forums by LDC. The BOLT parallel training consists of all of the high-quality NIST training, plus an additional 3 million words of translated forum data provided by LDC. The tuning and test sets consist of roughly 5000 segments each, with 2 references for Arabic and 3 for Chinese.

Results are shown in Table 4. The baseline here uses the same feature set as the strong NIST system. On Arabic, the total gain is +2.6 BLEU, while on Chinese, the gain is +1.3 BLEU.

BOLT Test		
	Ar-En	**Ch-En**
	BLEU	BLEU
Baseline (w/o RNNLM)	40.2	30.6
Baseline (w/ RNNLM)	41.3	30.9
+ S2T/L2R NNJM (Dec)	42.9	31.9
+ S2T NNLTM (Dec)	43.2	31.9
+ Other NNJMs (Resc)	43.9	32.2

Table 4: Primary results on Arabic-English and Chinese-English BOLT Web Forum. Each row includes the aggregate features from all previous rows.

6.4 Effect of k-best Rescoring Only

Table 5 shows performance when our S2T/L2R NNJM is used only in 1000-best rescoring, compared to decoding. The primary purpose of this is as a comparison to Le et al. (2012), whose model can only be used in k-best rescoring.

BOLT Test		
	Ar-En	
	Without RNNLM	**With RNNLM**
	BLEU	BLEU
Baseline	40.2	41.3
S2T/L2R NNJM (Resc)	41.7	41.6
S2T/L2R NNJM (Dec)	42.8	42.9

Table 5: Comparison of our primary NNJM in decoding vs. 1000-best rescoring.

We can see that the rescoring-only NNJM performs very well when used on top of a baseline without an RNNLM (+1.5 BLEU), but the gain on top of the RNNLM is very small (+0.3 BLEU). The gain from the decoding NNJM is large in both cases (+2.6 BLEU w/o RNNLM, +1.6 BLEU w/ RNNLM). This demonstrates that the full power of the NNJM can only be harnessed when it is used in decoding. It is also interesting to see that the RNNLM is no longer beneficial when the NNJM is used.

6.5 Effect of Neural Network Configuration

Table 6 shows results using the S2T/L2R NNJM with various configurations. We can see that reducing the source window size, layer size, or vocab size will all degrade results. Increasing the sizes beyond the default NNJM has almost no effect (102%). Also note that the target-only NNLM (i.e., Source Window=0) only obtains 33% of the improvements of the NNJM.

BOLT Test		
	Ar-En	
	BLEU	**% Gain**
"Simple Hier." Baseline	33.8	-
S2T/L2R NNJM (Dec)	38.4	100%
Source Window=7	38.3	98%
Source Window=5	38.2	96%
Source Window=3	37.8	87%
Source Window=0	35.3	33%
Layers=384x768x768	38.5	102%
Layers=192x512	38.1	93%
Layers=128x128	37.1	72%
Vocab=64,000	38.5	102%
Vocab=16,000	38.1	93%
Vocab=8,000	37.3	83%
Activation=Rectified Lin.	38.5	102%
Activation=Linear	37.3	76%

Table 6: Results with different neural network architectures. The "default" NNJM in the second row uses these parameters: SW=11, L=192x512x512, V=32,000, A=tanh. All models use a 3-word target history (i.e., 4-gram LM). "Layers" refers to the size of the word embedding followed by the hidden layers. "Vocab" refers to the size of the input and output vocabularies. "% Gain" is the BLEU gain over the baseline relative to the default NNJM.

6.6 Effect of Speedups

All previous results use a self-normalized neural network with two hidden layers. In Table 7, we compare this to using a standard network (with two hidden layers), as well as a pre-computed neural network.[13] The "Simple Hierarchical" baseline is used here because it more closely approximates a real-time MT engine. For the sake of speed, these experiments only use the S2T/L2R NNJM+S2T NNLTM.

[13]The difference in score for self-normalized vs. pre-computed is *entirely* due to two vs. one hidden layers.

Each result from Table 7 corresponds to a row in Table 2 of Section 2.4. We can see that going from the standard model to the pre-computed model only reduces the BLEU improvement from +6.4 to +6.1, while increasing the NNJM lookup speed by a factor of 10,000x.

BOLT Test		
	Ar-En	
	BLEU	**Gain**
"Simple Hier." Baseline	33.8	-
Standard NNJM	40.2	+6.4
Self-Norm NNJM	40.1	+6.3
Pre-Computed NNJM	39.9	+6.1

Table 7: Results for the standard NNs vs. self-normalized NNs vs. pre-computed NNs.

In Table 2 we showed that the cost of unique lookups for the pre-computed NNJM is only $\sim$0.001 seconds per source word. This does not include the cost of n-gram creation or cached lookups, which amount to $\sim$0.03 seconds per source word in our current implementation.[14] However, the n-grams created for the NNJM can be shared with the Kneser-Ney LM, which reduces the cost of that feature. Thus, the total cost *increase* of using the NNJM+NNLTM features in decoding is only $\sim$0.01 seconds per source word.

In future work we will provide more detailed analysis regarding the usability of the NNJM in a low-latency, high-throughput MT engine.

7 Related Work

Although there has been a substantial amount of past work in lexicalized joint models (Marino et al., 2006; Crego and Yvon, 2010), nearly all of these papers have used older statistical techniques such as Kneser-Ney or Maximum Entropy. However, not only are these techniques intractable to train with high-order context vectors, they also lack the neural network's ability to semantically generalize (Mikolov et al., 2013) and learn non-linear relationships.

A number of recent papers have proposed methods for creating neural network translation/joint models, but nearly all of these works have obtained much smaller BLEU improvements than ours. For each related paper, we will briefly con-

[14]In our decoder, roughly 95% of NNJM n-gram lookups within the same sentence are duplicates.

trast their methodology with our own and summarize their BLEU improvements using scores taken directly from the cited paper.

Auli et al. (2013) use a *fixed* continuous-space source representation, obtained from LDA (Blei et al., 2003) or a source-only NNLM. Also, their model is recurrent, so it cannot be used in decoding. They obtain +0.2 BLEU improvement on top of a target-only NNLM (25.6 vs. 25.8).

Schwenk (2012) predicts an entire target phrase at a time, rather than a word at a time. He obtains +0.3 BLEU improvement (24.8 vs. 25.1).

Zou et al. (2013) estimate context-free bilingual lexical similarity scores, rather than using a large context. They obtain an +0.5 BLEU improvement on Chinese-English (30.0 vs. 30.5).

Kalchbrenner and Blunsom (2013) implement a convolutional recurrent NNJM. They score a 1000-best list using only their model and are able to achieve the same BLEU as using all 12 standard MT features (21.8 vs 21.7). However, additive results are not presented.

The most similar work that we know of is Le et al. (2012). Le's basic procedure is to re-order the source to match the linear order of the target, and then segment the hypothesis into minimal bilingual phrase pairs. Then, he predicts each target word given the previous bilingual phrases. However, Le's formulation could only be used in k-best rescoring, since it requires long-distance reordering and a large target context.

Le's model does obtain an impressive +1.7 BLEU gain on top of a baseline without an NNLM (25.8 vs. 27.5). However, when compared to the strongest baseline which includes an NNLM, Le's best models (S2T + T2S) only obtain an +0.6 BLEU improvement (26.9 vs. 27.5). This is consistent with our rescoring-only result, which indicates that k-best rescoring is too shallow to take advantage of the power of a joint model.

Le's model also uses minimal phrases rather than being purely lexicalized, which has two main downsides: (a) a number of complex, hand-crafted heuristics are required to define phrase boundaries, which may not transfer well to new languages, (b) the effective vocabulary size is much larger, which substantially increases data sparsity issues.

We should note that our best results use six separate models, whereas all previous work only uses one or two models. However, we have demonstrated that we can obtain 50%-80% of the total improvement with only one model (S2T/L2R NNJM), and 70%-90% with only two models (S2T/L2R NNJM + S2T NNLTM). Thus, the one and two-model conditions still significantly outperform any past work.

8 Discussion

We have described a novel formulation for a neural network-based machine translation joint model, along with several simple variations of this model. When used as MT decoding features, these models are able to produce a gain of +3.0 BLEU on top of a very strong and feature-rich baseline, as well as a +6.3 BLEU gain on top of a simpler system.

Our model is remarkably simple – it requires no linguistic resources, no feature engineering, and only a handful of hyper-parameters. It also has no reliance on potentially fragile outside algorithms, such as unsupervised word clustering. We consider the simplicity to be a major advantage. Not only does this suggest that it will generalize well to new language pairs and domains, but it also suggests that it will be straightforward for others to replicate these results.

Overall, we believe that the following factors set us apart from past work and allowed us to obtain such significant improvements:

1. The ability to use the NNJM in decoding rather than rescoring.
2. The use of a large bilingual context vector, which is provided to the neural network in "raw" form, rather than as the output of some other algorithm.
3. The fact that the model is purely lexicalized, which avoids both data sparsity and implementation complexity.
4. The large size of the network architecture.
5. The directional variation models.

One of the biggest goals of this work is to quell any remaining doubts about the utility of neural networks in machine translation. We believe that there are large areas of research yet to be explored. For example, creating a new type of decoder centered around a purely lexicalized neural network model. Our short term ideas include using more interesting types of context in our input vector (such as source syntax), or using the NNJM to model syntactic/semantic structure of the target.

References

Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint language and translation modeling with recurrent neural networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1044–1054, Seattle, Washington, USA, October. Association for Computational Linguistics.

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

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March.

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *HLT-NAACL*, pages 218–226.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Josep Maria Crego and François Yvon. 2010. Factored bilingual *n*-gram language models for statistical machine translation. *Machine Translation*, 24(2):159–175.

Jacob Devlin and Spyros Matsoukas. 2012. Trait-based hypothesis selection for machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 528–532, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jacob Devlin. 2009. Lexical features for statistical machine translation. Master's thesis, University of Maryland.

Nizar Habash, Ryan Roth, Owen Rambow, Ramy Eskander, and Nadi Tomeh. 2013. Morphological analysis and disambiguation for dialectal arabic. In *HLT-NAACL*, pages 426–432.

Zhongqiang Huang, Jacob Devlin, and Rabih Zbib. 2013. Factored soft source syntactic constraints for hierarchical machine translation. In *EMNLP*, pages 556–566.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE.

Hai-Son Le, Alexandre Allauzen, and François Yvon. 2012. Continuous space translation models with neural networks. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 39–48, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yann LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. 1998. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–50. Springer.

José B Marino, Rafael E Banchs, Josep M Crego, Adrià De Gispert, Patrik Lambert, José AR Fonollosa, and Marta R Costa-Jussà. 2006. N-gram-based machine translation. *Computational Linguistics*, 32(4):527–549.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048.

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

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

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*.

Jason Riesa, Ann Irvine, and Daniel Marcu. 2011. Feature-rich language-independent syntax-based alignment for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 497–507, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ronald Rosenfeld. 1996. A maximum entropy approach to adaptive statistical language modeling. *Computer, Speech and Language*, 10:187–228.

Antti Rosti, Bing Zhang, Spyros Matsoukas, and Rich Schwartz. 2010. BBN system description for WMT10 system combination task. In *WMT/MetricsMATR*, pages 321–326.

Holger Schwenk. 2010. Continuous-space language models for statistical machine translation. *Prague Bull. Math. Linguistics*, 93:137–146.

Holger Schwenk. 2012. Continuous space translation models for phrase-based statistical machine translation. In *COLING (Posters)*, pages 1071–1080.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2010. String-to-dependency statistical machine translation. *Computational Linguistics*, 36(4):649–671, December.

Matthew Snover, Bonnie Dorr, and Richard Schwartz. 2008. Language and translation model adaptation using comparable corpora. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 857–866, Stroudsburg, PA, USA. Association for Computational Linguistics.

Makoto Tanaka, Yasuhara Toru, Jun-ya Yamamoto, and Mikio Norimatsu. 2013. An efficient language model using double-array structures.

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

Will Y Zou, Richard Socher, Daniel Cer, and Christopher D Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1393–1398.

Low-Rank Tensors for Scoring Dependency Structures

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{taolei, yuxin, yuanzh, regina, tommi}@csail.mit.edu

Abstract

Accurate scoring of syntactic structures such as head-modifier arcs in dependency parsing typically requires rich, high-dimensional feature representations. A small subset of such features is often selected manually. This is problematic when features lack clear linguistic meaning as in embeddings or when the information is blended across features. In this paper, we use tensors to map high-dimensional feature vectors into low dimensional representations. We explicitly maintain the parameters as a low-rank tensor to obtain low dimensional representations of words in their syntactic roles, and to leverage modularity in the tensor for easy training with online algorithms. Our parser consistently outperforms the Turbo and MST parsers across 14 different languages. We also obtain the best published UAS results on 5 languages.[1]

1 Introduction

Finding an expressive representation of input sentences is crucial for accurate parsing. Syntactic relations manifest themselves in a broad range of surface indicators, ranging from morphological to lexical, including positional and part-of-speech (POS) tagging features. Traditionally, parsing research has focused on modeling the direct connection between the features and the predicted syntactic relations such as head-modifier (arc) relations in dependency parsing. Even in the case of first-order parsers, this results in a high-dimensional vector representation of each arc. Discrete features, and their cross products, can be further complemented with auxiliary information about words participating in an arc, such as continuous vector representations of words. The exploding dimensionality of rich feature vectors must then be balanced with the difficulty of effectively learning the associated parameters from limited training data.

A predominant way to counter the high dimensionality of features is to manually design or select a meaningful set of feature templates, which are used to generate different types of features (McDonald et al., 2005a; Koo and Collins, 2010; Martins et al., 2013). Direct manual selection may be problematic for two reasons. First, features may lack clear linguistic interpretation as in distributional features or continuous vector embeddings of words. Second, designing a small subset of templates (and features) is challenging when the relevant linguistic information is distributed across the features. For instance, morphological properties are closely tied to part-of-speech tags, which in turn relate to positional features. These features are not redundant. Therefore, we may suffer a performance loss if we select only a small subset of the features. On the other hand, by including all the rich features, we face over-fitting problems.

We depart from this view and leverage high-dimensional feature vectors by mapping them into low dimensional representations. We begin by representing high-dimensional feature vectors as multi-way cross-products of smaller feature vectors that represent words and their syntactic relations (arcs). The associated parameters are viewed as a tensor (multi-way array) of low rank, and optimized for parsing performance. By explicitly representing the tensor in a low-rank form, we have direct control over the effective dimensionality of the set of parameters. We obtain role-dependent low-dimensional representations for words (head, modifier) that are specifically tailored for parsing accuracy, and use standard online algorithms for optimizing the low-rank tensor components.

The overall approach has clear linguistic and

[1] Our code is available at https://github.com/taolei87/RBGParser.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1381–1391,
Baltimore, Maryland, USA, June 23-25 2014.

computational advantages:

- Our low dimensional embeddings are tailored to the syntactic context of words (head, modifier). This low dimensional syntactic abstraction can be thought of as a proxy to manually constructed POS tags.
- By automatically selecting a small number of dimensions useful for parsing, we can leverage a wide array of (correlated) features. Unlike parsers such as MST, we can easily benefit from auxiliary information (e.g., word vectors) appended as features.

We implement the low-rank factorization model in the context of first- and third-order dependency parsing. The model was evaluated on 14 languages, using dependency data from CoNLL 2008 and CoNLL 2006. We compare our results against the MST (McDonald et al., 2005a) and Turbo (Martins et al., 2013) parsers. The low-rank parser achieves average performance of 89.08% across 14 languages, compared to 88.73% for the Turbo parser, and 87.19% for MST. The power of the low-rank model becomes evident in the absence of any part-of-speech tags. For instance, on the English dataset, the low-rank model trained without POS tags achieves 90.49% on first-order parsing, while the baseline gets 86.70% if trained under the same conditions, and 90.58% if trained with 12 core POS tags. Finally, we demonstrate that the model can successfully leverage word vector representations, in contrast to the baselines.

2 Related Work

Selecting Features for Dependency Parsing A great deal of parsing research has been dedicated to feature engineering (Lazaridou et al., 2013; Marton et al., 2010; Marton et al., 2011). While in most state-of-the-art parsers, features are selected manually (McDonald et al., 2005a; McDonald et al., 2005b; Koo and Collins, 2010; Martins et al., 2013; Zhang and McDonald, 2012a; Rush and Petrov, 2012a), automatic feature selection methods are gaining popularity (Martins et al., 2011b; Ballesteros and Nivre, 2012; Nilsson and Nugues, 2010; Ballesteros, 2013). Following standard machine learning practices, these algorithms iteratively select a subset of features by optimizing parsing performance on a development set. These feature selection methods are particularly promising in parsing scenarios where the optimal feature set is likely to be a small subset of the original set of candidate features. Our technique, in contrast, is suitable for cases where the relevant information is distributed across a larger set of related features.

Embedding for Dependency Parsing A lot of recent work has been done on mapping words into vector spaces (Collobert and Weston, 2008; Turian et al., 2010; Dhillon et al., 2011; Mikolov et al., 2013). Traditionally, these vector representations have been derived primarily from co-occurrences of words within sentences, ignoring syntactic roles of the co-occurring words. Nevertheless, any such word-level representation can be used to offset inherent sparsity problems associated with full lexicalization (Cirik and Şensoy, 2013). In this sense they perform a role similar to POS tags.

Word-level vector space embeddings have so far had limited impact on parsing performance. From a computational perspective, adding non-sparse vectors directly as features, including their combinations, can significantly increase the number of active features for scoring syntactic structures (e.g., dependency arc). Because of this issue, Cirik and Şensoy (2013) used word vectors only as unigram features (without combinations) as part of a shift reduce parser (Nivre et al., 2007). The improvement on the overall parsing performance was marginal. Another application of word vectors is compositional vector grammar (Socher et al., 2013). While this method learns to map word combinations into vectors, it builds on existing word-level vector representations. In contrast, we represent words as vectors in a manner that is directly optimized for parsing. This framework enables us to learn new syntactically guided embeddings while also leveraging separately estimated word vectors as starting features, leading to improved parsing performance.

Dimensionality Reduction Many machine learning problems can be cast as matrix problems where the matrix represents a set of co-varying parameters. Such problems include, for example, multi-task learning and collaborative filtering. Rather than assuming that each parameter can be set independently of others, it is helpful to assume that the parameters vary in a low dimensional subspace that has to be estimated together with the parameters. In terms of the parameter matrix, this corresponds to a low-rank assumption. Low-rank constraints are commonly used for improving

generalization (Lee and Seung, 1999; Srebro et al., 2003; Srebro et al., 2004; Evgeniou and Pontil, 2007)

A strict low-rank assumption can be restrictive. Indeed, recent approaches to matrix problems decompose the parameter matrix as a sum of low-rank and sparse matrices (Tao and Yuan, 2011; Zhou and Tao, 2011). The sparse matrix is used to highlight a small number of parameters that should vary independently even if most of them lie on a low-dimensional subspace (Waters et al., 2011; Chandrasekaran et al., 2011). We follow this decomposition while extending the parameter matrix into a tensor.

Tensors are multi-way generalizations of matrices and possess an analogous notion of rank. Tensors are increasingly used as tools in spectral estimation (Hsu and Kakade, 2013), including in parsing (Cohen et al., 2012) and other NLP problems (de Cruys et al., 2013), where the goal is to avoid local optima in maximum likelihood estimation. In contrast, we expand features for parsing into a multi-way tensor, and operate with an explicit low-rank representation of the associated parameter tensor. The explicit representation sidesteps inherent complexity problems associated with the tensor rank (Hillar and Lim, 2009). Our parameters are divided into a sparse set corresponding to manually chosen MST or Turbo parser features and a larger set governed by a low-rank tensor.

3 Problem Formulation

We will commence here by casting first-order dependency parsing as a tensor estimation problem. We will start by introducing the notation used in the paper, followed by a more formal description of our dependency parsing task.

3.1 Basic Notations

Let $A \in \mathbb{R}^{n \times n \times d}$ be a 3-dimensional tensor (a 3-way array). We denote each element of the tensor as $A_{i,j,k}$ where $i \in [n], j \in [n], k \in [d]$ and $[n]$ is a shorthand for the set of integers $\{1, 2, \cdots, n\}$. Similarly, we use $M_{i,j}$ and u_i to represent the elements of matrix M and vector u, respectively.

We define the *inner product* of two tensors (or matrices) as $\langle A, B \rangle = \text{vec}(A)^T \text{vec}(B)$, where $\text{vec}(\cdot)$ concatenates the tensor (or matrix) elements into a column vector. The *squared norm* of a tensor/matrix is denoted by $\|A\|^2 = \langle A, A \rangle$.

The *Kronecker product* of three vectors is denoted by $u \otimes v \otimes w$ and forms a rank-1 tensor such that

$$(u \otimes v \otimes w)_{i,j,k} = u_i v_j w_k.$$

Note that the vectors u, v, and w may be column or row vectors. Their orientation is defined based on usage. For example, $u \otimes v$ is a rank-1 matrix uv^T when u and v are column vectors ($u^T v$ if they are row vectors).

We say that tensor A is in Kruskal form if

$$A = \sum_{i=1}^{r} U(i,:) \otimes V(i,:) \otimes W(i,:) \qquad (1)$$

where $U, V \in \mathbb{R}^{r \times n}$, $W \in \mathbb{R}^{r \times d}$ and $U(i,:)$ is the i^{th} row of matrix U. We will directly learn a low-rank tensor A (because r is small) in this form as one of our model parameters.

3.2 Dependency Parsing

Let x be a sentence and $\mathcal{Y}(x)$ the set of possible dependency trees over the words in x. We assume that the score $S(x, y)$ of each candidate dependency tree $y \in \mathcal{Y}(x)$ decomposes into a sum of "local" scores for arcs. Specifically:

$$S(x, y) = \sum_{h \to m \in y} s(h \to m) \qquad \forall y \in \mathcal{Y}(x)$$

where $h \to m$ is the head-modifier dependency arc in the tree y. Each y is understood as a collection of arcs $h \to m$ where h and m index words in x.[2] For example, $x(h)$ is the word corresponding to h. We suppress the dependence on x whenever it is clear from context. For example, $s(h \to m)$ can depend on x in complicated ways as discussed below. The predicted parse is obtained as $\hat{y} = \arg\max_{y \in \mathcal{Y}(x)} S(x, y)$.

A key problem is how we parameterize the arc scores $s(h \to m)$. Following the MST parser (McDonald et al., 2005a) we can define rich features characterizing each head-modifier arc, compiled into a sparse binary vector $\phi_{h \to m} \in \mathbb{R}^L$ that depends on the sentence x as well as the chosen arc $h \to m$ (again, we suppress the dependence on x). Based on this feature representation, we define the score of each arc as $s_\theta(h \to m) =$

[2] Note that in the case of high-order parsing, the sum $S(x, y)$ may also include local scores for other syntactic structures, such as grandhead-head-modifier score $s(g \to h \to m)$. See (Martins et al., 2013) for a complete list of these structures.

Unigram features:		
form	form-p	form-n
lemma	lemma-p	lemma-n
pos	pos-p	pos-n
morph	bias	
Bigram features:		
pos-p, pos		
pos, pos-n		
pos, lemma		
morph, lemma		
Trigram features:		
pos-p, pos, pos-n		

Table 1: Word feature templates used by our model. pos, form, lemma and morph stand for the fine POS tag, word form, word lemma and the morphology feature (provided in CoNLL format file) of the current word. There is a bias term that is always active for any word. The suffixes -p and -n refer to the left and right of the current word respectively. For example, pos-p means the POS tag to the left of the current word in the sentence.

$\langle \theta, \phi_{h \to m} \rangle$ where $\theta \in \mathbb{R}^L$ represent adjustable parameters to be learned, and L is the number of parameters (and possible features in $\phi_{h \to m}$).

We can alternatively specify arc features in terms of rank-1 tensors by taking the Kronecker product of simpler feature vectors associated with the head (vector $\phi_h \in \mathbb{R}^n$), and modifier (vector $\phi_m \in \mathbb{R}^n$), as well as the arc itself (vector $\phi_{h,m} \in \mathbb{R}^d$). Here $\phi_{h,m}$ is much lower dimensional than the MST arc feature vector $\phi_{h \to m}$ discussed earlier. For example, $\phi_{h,m}$ may be composed of only indicators for binned arc lengths[3]. ϕ_h and ϕ_m, on the other hand, are built from features shown in Table 1. By taking the cross-product of all these component feature vectors, we obtain the full feature representation for arc $h \to m$ as a rank-1 tensor

$$\phi_h \otimes \phi_m \otimes \phi_{h,m} \in \mathbb{R}^{n \times n \times d}$$

Note that elements of this rank-1 tensor include feature combinations that are not part of the feature crossings in $\phi_{h \to m}$. In this sense, the rank-1 tensor represents a substantial feature expansion. The arc score $s_{tensor}(h \to m)$ associated with the tensor representation is defined analogously as

$$s_{tensor}(h \to m) = \langle A, \phi_h \otimes \phi_m \otimes \phi_{h,m} \rangle$$

where the adjustable parameters A also form a tensor. Given the typical dimensions of the component feature vectors, ϕ_h, ϕ_m, $\phi_{h,m}$, it is not even possible to store all the parameters in A. Indeed, in the full English training set of CoNLL-2008, the tensor involves around 8×10^{11} entries while the MST feature vector has approximately 1.5×10^7 features. To counter this feature explosion, we restrict the parameters A to have low rank.

Low-Rank Dependency Scoring We can represent a rank-r tensor A explicitly in terms of parameter matrices U, V, and W as shown in Eq. 1. As a result, the arc score for the tensor reduces to evaluating $U\phi_h$, $V\phi_m$, and $W\phi_{h,m}$ which are all r dimensional vectors and can be computed efficiently based on any sparse vectors ϕ_h, ϕ_m, and $\phi_{h,m}$. The resulting arc score $s_{tensor}(h \to m)$ is then

$$\sum_{i=1}^{r} [U\phi_h]_i [V\phi_m]_i [W\phi_{h,m}]_i \qquad (2)$$

By learning parameters U, V, and W that function well in dependency parsing, we also learn context-dependent embeddings for words and arcs. Specifically, $U\phi_h$ (for a given sentence, suppressed) is an r dimensional vector representation of the word corresponding to h as a head word. Similarly, $V\phi_m$ provides an analogous representation for a modifier m. Finally, $W\phi_{h,m}$ is a vector embedding of the supplemental arc-dependent information. The resulting embedding is therefore tied to the syntactic roles of the words (and arcs), and learned in order to perform well in parsing.

We expect a dependency parsing model to benefit from several aspects of the low-rank tensor scoring. For example, we can easily incorporate additional useful features in the feature vectors ϕ_h, ϕ_m and $\phi_{h,m}$, since the low-rank assumption (for small enough r) effectively counters the otherwise uncontrolled feature expansion. Moreover, by controlling the amount of information we can extract from each of the component feature vectors (via rank r), the statistical estimation problem does not scale dramatically with the dimensions of ϕ_h, ϕ_m and $\phi_{h,m}$. In particular, the low-rank constraint can help generalize to unseen arcs. Consider a feature $\delta(x(h) = a) \cdot \delta(x(m) =$

[3] In our current version, $\phi_{h,m}$ only contains the binned arc length. Other possible features include, for example, the label of the arc $h \to m$, the POS tags between the head and the modifier, boolean flags which indicate the occurence of in-between punctutations or conjunctions, etc.

$b) \cdot \delta(dis(x,h,m) = c)$ which is non-zero only for an arc $a \rightarrow b$ with distance c in sentence x. If the arc has not been seen in the available training data, it does not contribute to the traditional arc score $s_\theta(\cdot)$. In contrast, with the low-rank constraint, the arc score in Eq. 2 would typically be non-zero.

Combined Scoring Our parsing model aims to combine the strengths of both traditional features from the MST/Turbo parser as well as the new low-rank tensor features. In this way, our model is able to capture a wide range of information including the auxiliary features without having uncontrolled feature explosion, while still having the full accessibility to the manually engineered features that are proven useful. Specifically, we define the arc score $s_\gamma(h \rightarrow m)$ as the combination

$$\begin{aligned}
&(1-\gamma) s_{tensor}(h \rightarrow m) + \gamma s_\theta(h \rightarrow m) \\
&= (1-\gamma) \sum_{i=1}^{r} [U\phi_h]_i [V\phi_m]_i [W\phi_{h,m}]_i \\
&+ \gamma \langle \theta, \phi_{h \rightarrow m} \rangle \qquad (3)
\end{aligned}$$

where $\theta \in \mathbb{R}^L$, $U \in \mathbb{R}^{r \times n}$, $V \in \mathbb{R}^{r \times n}$, and $W \in \mathbb{R}^{r \times d}$ are the model parameters to be learned. The rank r and $\gamma \in [0,1]$ (balancing the two scores) represent hyper-parameters in our model.

4 Learning

The training set $D = \{(\hat{x}_i, \hat{y}_i)\}_{i=1}^N$ consists of N pairs, where each pair consists of a sentence x_i and the corresponding gold (target) parse y_i. The goal is to learn values for the parameters θ, U, V and W that optimize the combined scoring function $S_\gamma(x,y) = \sum_{h \rightarrow m \in y} s_\gamma(h \rightarrow m)$, defined in Eq. 3, for parsing performance. We adopt a maximum soft-margin framework for this learning problem. Specifically, we find parameters θ, U, V, W, and $\{\xi_i\}$ that minimize

$$\begin{aligned}
&C \sum_i \xi_i + \|\theta\|^2 + \|U\|^2 + \|V\|^2 + \|W\|^2 \\
&\text{s.t.} \quad S_\gamma(\hat{x}_i, \hat{y}_i) \geq S_\gamma(\hat{x}_i, y_i) + \|\hat{y}_i - y_i\|_1 - \xi_i \\
&\qquad \forall y_i \in \mathcal{Y}(\hat{x}_i),\ \forall i. \qquad (4)
\end{aligned}$$

where $\|\hat{y}_i - y_i\|_1$ is the number of mismatched arcs between the two trees, and ξ_i is a non-negative slack variable. The constraints serve to separate the gold tree from other alternatives in $\mathcal{Y}(\hat{x}_i)$ with a margin that increases with distance.

The objective as stated is not jointly convex with respect to U, V and W due to our explicit representation of the low-rank tensor. However, if we fix any two sets of parameters, for example, if we fix V and W, then the combined score $S_\gamma(x,y)$ will be a linear function of both θ and U. As a result, the objective will be jointly convex with respect to θ and U and could be optimized using standard tools. However, to accelerate learning, we adopt an online learning setup. Specifically, we use the passive-aggressive learning algorithm (Crammer et al., 2006) tailored to our setting, updating pairs of parameter sets, (θ, U), (θ, V) and (θ, W) in an alternating manner. This method is described below.

Online Learning In an online learning setup, we update parameters successively based on each sentence. In order to apply the passive-aggressive algorithm, we fix two of U, V and W (say, for example, V and W) in an alternating manner, and apply a closed-form update to the remaining parameters (here U and θ). This is possible since the objective function with respect to (θ, U) has a similar form as in the original passive-aggressive algorithm. To illustrate this, consider a training sentence x_i. The update involves finding first the best competing tree,

$$\tilde{y}_i = \arg\max_{y_i \in \mathcal{Y}(\hat{x}_i)} S_\gamma(\hat{x}_i, y_i) + \|\hat{y}_i - y_i\|_1 \qquad (5)$$

which is the tree that violates the constraint in Eq. 4 most (i.e. maximizes the loss ξ_i). We then obtain parameter increments $\Delta\theta$ and ΔU by solving

$$\begin{aligned}
&\min_{\Delta\theta,\ \Delta U,\ \xi \geq 0} \quad \frac{1}{2}\|\Delta\theta\|^2 + \frac{1}{2}\|\Delta U\|^2 + C\xi \\
&\text{s.t.} \quad S_\gamma(\hat{x}_i, \hat{y}_i) \geq S_\gamma(\hat{x}_i, \tilde{y}_i) + \|\hat{y}_i - \tilde{y}_i\|_1 - \xi
\end{aligned}$$

In this way, the optimization problem attempts to keep the parameter change as small as possible, while forcing it to achieve mostly zero loss on this single instance. This problem has a closed form solution

$$\begin{aligned}
\Delta\theta &= \min\left\{C, \frac{\text{loss}}{\gamma^2\|d\theta\|^2 + (1-\gamma)^2\|du\|^2}\right\} \gamma d\theta \\
\Delta U &= \min\left\{C, \frac{\text{loss}}{\gamma^2\|d\theta\|^2 + (1-\gamma)^2\|du\|^2}\right\} (1-\gamma) du
\end{aligned}$$

where

$$\begin{aligned}
\text{loss} &= S_\gamma(\hat{x}_i, \tilde{y}_i) + \|\hat{y}_i - \tilde{y}_i\|_1 - S_\gamma(\hat{x}_i, \hat{y}_i) \\
d\theta &= \sum_{h \to m \in \hat{y}_i} \phi_{h \to m} - \sum_{h \to m \in \tilde{y}_i} \phi_{h \to m} \\
du &= \sum_{h \to m \in \hat{y}_i} [(V\phi_m) \odot (W\phi_{h,m})] \otimes \phi_h \\
&- \sum_{h \to m \in \tilde{y}_i} [(V\phi_m) \odot (W\phi_{h,m})] \otimes \phi_h
\end{aligned}$$

where $(u \odot v)_i = u_i v_i$ is the Hadamard (element-wise) product. The magnitude of change of θ and U is controlled by the parameter C. By varying C, we can determine an appropriate step size for the online updates. The updates also illustrate how γ balances the effect of the MST component of the score relative to the low-rank tensor score. When $\gamma = 0$, the arc scores are entirely based on the low-rank tensor and $\Delta\theta = 0$. Note that ϕ_h, ϕ_m, $\phi_{h,m}$, and $\phi_{h \to m}$ are typically very sparse for each word or arc. Therefore du and $d\theta$ are also sparse and can be computed efficiently.

Initialization The alternating online algorithm relies on how we initialize U, V, and W since each update is carried out in the context of the other two. A random initialization of these parameters is unlikely to work well, both due to the dimensions involved, and the nature of the alternating updates. We consider here instead a reasonable deterministic "guess" as the initialization method.

We begin by training our model without any low-rank parameters, and obtain parameters θ. The majority of features in this MST component can be expressed as elements of the feature tensor, i.e., as $[\phi_h \otimes \phi_m \otimes \phi_{h,m}]_{i,j,k}$. We can therefore create a tensor representation of θ such that $B_{i,j,k}$ equals the corresponding parameter value in θ. We use a low-rank version of B as the initialization. Specifically, we unfold the tensor B into a matrix $B^{(h)}$ of dimensions n and nd, where $n = dim(\phi_h) = dim(\phi_m)$ and $d = dim(\phi_{h,m})$. For instance, a rank-1 tensor can be unfolded as $u \otimes v \otimes w = u \otimes \text{vec}(v \otimes w)$. We compute the top-r SVD of the resulting unfolded matrix such that $B^{(h)} = P^T S Q$. U is initialized as P. Each right singular vector $S_i Q(i,:)$ is also a matrix in $\mathbb{R}^{n \times d}$. The leading left and right singular vectors of this matrix are assigned to $V(i,:)$ and $W(i,:)$ respectively. In our implementation, we run one epoch of our model without low-rank parameters and initialize the tensor A.

Parameter Averaging The passive-aggressive algorithm regularizes the increments (e.g. $\Delta\theta$ and ΔU) during each update but does not include any overall regularization. In other words, keeping updating the model may lead to large parameter values and over-fitting. To counter this effect, we use parameter averaging as used in the MST and Turbo parsers. The final parameters are those averaged across all the iterations (cf. (Collins, 2002)). For simplicity, in our algorithm we average U, V, W and θ separately, which works well empirically.

5 Experimental Setup

Datasets We test our dependency model on 14 languages, including the English dataset from CoNLL 2008 shared tasks and all 13 datasets from CoNLL 2006 shared tasks (Buchholz and Marsi, 2006; Surdeanu et al., 2008). These datasets include manually annotated dependency trees, POS tags and morphological information. Following standard practices, we encode this information as features.

Methods We compare our model to MST and Turbo parsers on non-projective dependency parsing. For our parser, we train both a first-order parsing model (as described in Section 3 and 4) as well as a third-order model. The third order parser simply adds high-order features, those typically used in MST and Turbo parsers, into our $s_\theta(x, y) = \langle \theta, \phi(x, y) \rangle$ scoring component. The decoding algorithm for the third-order parsing is based on (Zhang et al., 2014). For the Turbo parser, we directly compare with the recent published results in (Martins et al., 2013). For the MST parser, we train and test using the most recent version of the code.[4] In addition, we implemented two additional baselines, NT-1st (first order) and NT-3rd (third order), corresponding to our model without the tensor component.

Features For the arc feature vector $\phi_{h \to m}$, we use the same set of feature templates as MST v0.5.1. For head/modifier vector ϕ_h and ϕ_m, we show the complete set of feature templates used by our model in Table 1. Finally, we use a similar set of feature templates as Turbo v2.1 for 3rd order parsing.

To add auxiliary word vector representations, we use the publicly available word vectors (Cirik

[4] http://sourceforge.net/projects/mstparser/

	First-order only				High-order				
	Ours	NT-1st	MST	Turbo	Ours-3rd	NT-3rd	MST-2nd	Turbo-3rd	Best Published
Arabic	79.60	78.71	78.3	77.23	79.95	79.53	78.75	79.64	81.12 (Ma11)
Bulgarian	92.30	91.14	90.98	91.76	93.50	92.79	91.56	93.1	94.02 (Zh13)
Chinese	91.43	90.85	90.40	88.49	**92.68**	92.39	91.77	89.98	91.89 (Ma10)
Czech	87.90	86.62	86.18	87.66	**90.50**	89.43	87.3	90.32	90.32 (Ma13)
Danish	90.64	89.80	89.84	89.42	91.39	90.82	90.5	91.48	92.00 (Zh13)
Dutch	84.81	83.77	82.89	83.61	**86.41**	86.08	84.11	86.19	86.19 (Ma13)
English	91.84	91.40	90.59	91.21	93.02	92.82	91.54	93.22	93.22 (Ma13)
German	90.24	89.70	89.54	90.52	91.97	92.26	90.14	92.41	92.41 (Ma13)
Japanese	**93.74**	93.36	93.38	92.78	93.71	93.23	92.92	93.52	93.72 (Ma11)
Portuguese	90.94	90.67	89.92	91.14	91.92	91.63	91.08	92.69	93.03 (Ko10)
Slovene	84.25	83.15	82.09	82.81	86.24	86.07	83.25	86.01	86.95 (Ma11)
Spanish	85.27	84.95	83.79	83.61	**88.00**	87.47	84.33	85.59	87.96 (Zh13)
Swedish	89.86	89.66	88.27	89.36	91.00	90.83	89.05	91.14	91.62 (Zh13)
Turkish	75.84	74.89	74.81	75.98	76.84	75.83	74.39	76.9	77.55 (Ko10)
Average	87.76	87.05	86.5	86.83	89.08	88.66	87.19	88.73	89.43

Table 2: First-order parsing (left) and high-order parsing (right) results on CoNLL-2006 datasets and the English dataset of CoNLL-2008. For our model, the experiments are ran with rank $r = 50$ and hyper-parameter $\gamma = 0.3$. To remove the tensor in our model, we ran experiments with $\gamma = 1$, corresponding to columns NT-1st and NT-3rd. The last column shows results of most accurate parsers among Nivre et al. (2006), McDonald et al. (2006), Martins et al. (2010), Martins et al. (2011a), Martins et al. (2013), Koo et al. (2010), Rush and Petrov (2012b), Zhang and McDonald (2012b) and Zhang et al. (2013).

and Şensoy, 2013), learned from raw data (Globerson et al., 2007; Maron et al., 2010). Three languages in our dataset – English, German and Swedish – have corresponding word vectors in this collection.[5] The dimensionality of this representation varies by language: English has 50 dimensional word vectors, while German and Swedish have 25 dimensional word vectors. Each entry of the word vector is added as a feature value into feature vectors ϕ_h and ϕ_m. For each word in the sentence, we add its own word vector as well as the vectors of its left and right words.

We should note that since our model parameter A is represented and learned in the low-rank form, we only have to store and maintain the low-rank projections $U\phi_h$, $V\phi_m$ and $W\phi_{h,m}$ rather than explicitly calculate the feature tensor $\phi_h \otimes \phi_m \otimes \phi_{h,m}$. Therefore updating parameters and decoding a sentence is still efficient, i.e., linear in the number of values of the feature vector. In contrast, assume we take the cross-product of the auxiliary word vector values, POS tags and lexical items of a word and its context, and add the crossed values into a normal model (in $\phi_{h \to m}$). The number of features for each arc would be at least quadratic, growing into thousands, and would be a significant impediment to parsing efficiency.

Evaluation Following standard practices, we train our full model and the baselines for 10 epochs. As the evaluation measure, we use unlabeled attachment scores (UAS) excluding punctuation. In all the reported experiments, the hyper-parameters are set as follows: $r = 50$ (rank of the tensor), $C = 1$ for first-order model and $C = 0.01$ for third-order model.

[5] https://github.com/wolet/sprml13-word-embeddings

6 Results

Overall Performance Table 2 shows the performance of our model and the baselines on 14 CoNLL datasets. Our model outperforms Turbo parser, MST parser, as well as its own variants without the tensor component. The improvements of our low-rank model are consistent across languages: results for the first order parser are better on 11 out of 14 languages. By comparing NT-1st and NT-3rd (models without low-rank) with our full model (with low-rank), we obtain 0.7% absolute improvement on first-order parsing, and 0.3% improvement on third-order parsing. Our model also achieves the best UAS on 5 languages.

We next focus on the first-order model and gauge the impact of the tensor component. First, we test our model by varying the hyper-parameter γ which balances the tensor score and the traditional MST/Turbo score components. Figure 1 shows the average UAS on CoNLL test datasets after each training epoch. We can see that the improvement of adding the low-rank tensor is consistent across various choices of hyper parame-

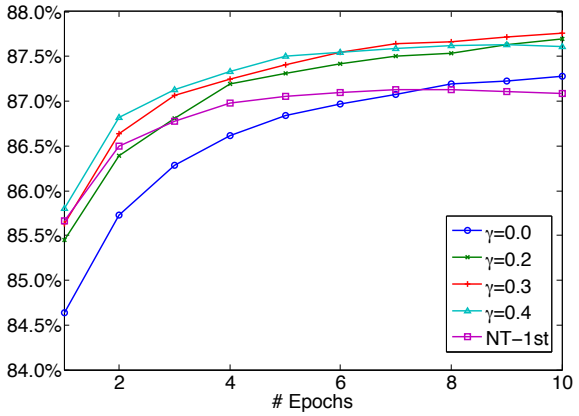


Figure 1: Average UAS on CoNLL testsets after different epochs. Our full model consistently performs better than NT-1st (its variation without tensor component) under different choices of the hyper-parameter γ.

	no word vector	with word vector
English	91.84	92.07
German	90.24	90.48
Swedish	89.86	90.38

Table 3: Results of adding unsupervised word vectors to the tensor. Adding this information yields consistent improvement for all languages.

ter γ. When training with the tensor component alone ($\gamma = 0$), the model converges more slowly. Learning of the tensor is harder because the scoring function is not linear (nor convex) with respect to parameters U, V and W. However, the tensor scoring component achieves better generalization on the test data, resulting in better UAS than NT-1st after 8 training epochs.

To assess the ability of our model to incorporate a range of features, we add unsupervised word vectors to our model. As described in previous section, we do so by appending the values of different coordinates in the word vector into ϕ_h and ϕ_m. As Table 3 shows, adding this information increases the parsing performance for all the three languages. For instance, we obtain more than 0.5% absolute improvement on Swedish.

Syntactic Abstraction without POS Since our model learns a compressed representation of feature vectors, we are interested to measure its performance when part-of-speech tags are not provided (See Table 4). The rationale is that given all other features, the model would induce representations that play a similar role to POS tags. Note that

	Our model		NT-1st	
	-POS	+wv.	-POS	+POS
English	88.89	90.49	86.70	90.58
German	82.63	85.80	78.71	88.50
Swedish	81.84	85.90	79.65	88.75

Table 4: The first three columns show parsing results when models are trained without POS tags. The last column gives the upper-bound, i.e. the performance of a parser trained with **12 Core POS tags**. The low-rank model outperforms NT-1st by a large margin. Adding word vector features further improves performance.

the performance of traditional parsers drops when tags are not provided. For example, the performance gap is 10% on German. Our experiments show that low-rank parser operates effectively in the absence of tags. In fact, it nearly reaches the performance of the original parser that used the tags on English.

Examples of Derived Projections We manually analyze low-dimensional projections to assess whether they capture syntactic abstraction. For this purpose, we train a model with only a tensor component (such that it has to learn an accurate tensor) on the English dataset and obtain low dimensional embeddings $U\phi_w$ and $V\phi_w$ for each word. The two r-dimension vectors are concatenated as an "averaged" vector. We use this vector to calculate the cosine similarity between words. Table 5 shows examples of five closest neighbors of queried words. While these lists include some noise, we can clearly see that the neighbors exhibit similar syntactic behavior. For example, "on" is close to other prepositions. More interestingly, we can consider the impact of syntactic context on the derived projections. The bottom part of Table 5 shows that the neighbors change substantially depending on the syntactic role of the word. For example, the closest words to the word "increase" are verbs in the context phrase "will increase again", while the closest words become nouns given a different phrase "an increase of".

Running Time Table 6 illustrates the impact of estimating low-rank tensor parameters on the running time of the algorithm. For comparison, we also show the NT-1st times across three typical languages. The Arabic dataset has the longest average sentence length, while the Chinese dataset

greatly	**profit**	**says**	**on**	**when**
actively	earnings	adds	with	where
openly	franchisees	predicts	into	what
significantly	shares	noted	at	why
outright	revenue	wrote	during	which
substantially	members	contends	over	who

increase	will **increase** again	an **increase** of
rise	arguing	gain
advance	be	prices
contest	charging	payment
halt	gone	members
Exchequer	making	subsidiary

hit	attacks **hit** the	hardest **hit** is
shed	distributes	monopolies
rallied	stayed	pills
triggered	sang	sophistication
appeared	removed	ventures
understate	eased	factors

Table 5: Five closest neighbors of the queried words (shown in bold). The upper part shows our learned embeddings group words with similar syntactic behavior. The two bottom parts of the table demonstrate that how the projections change depending on the syntactic context of the word.

	#Tok.	Len.	Train. Time (hour)	
			NT-1st	Ours
Arabic	42K	32	0.13	0.22
Chinese	337K	6	0.37	0.65
English	958K	24	1.88	2.83

Table 6: Comparison of training times across three typical datasets. The second column is the number of tokens in each data set. The third column shows the average sentence length. Both first-order models are implemented in Java and run as a single process.

has the shortest sentence length in CoNLL 2006. Based on these results, estimating a rank-50 tensor together with MST parameters only increases the running time by a factor of 1.7.

7 Conclusions

Accurate scoring of syntactic structures such as head-modifier arcs in dependency parsing typically requires rich, high-dimensional feature representations. We introduce a low-rank factorization method that enables to map high dimensional feature vectors into low dimensional representations. Our method maintains the parameters as a low-rank tensor to obtain low dimensional representations of words in their syntactic roles, and to leverage modularity in the tensor for easy training with online algorithms. We implement the approach on first-order to third-order dependency parsing. Our parser outperforms the Turbo and MST parsers across 14 languages.

Future work involves extending the tensor component to capture higher-order structures. In particular, we would consider second-order structures such as grandparent-head-modifier by increasing the dimensionality of the tensor. This tensor will accordingly be a four or five-way array. The online update algorithm remains applicable since each dimension is optimized in an alternating fashion.

8 Acknowledgements

The authors acknowledge the support of the MURI program (W911NF-10-1-0533) and the DARPA BOLT program. This research is developed in collaboration with the Arabic Language Technoligies (ALT) group at Qatar Computing Research Institute (QCRI) within the LYAS project. We thank Volkan Cirik for sharing the unsupervised word vector data. Thanks to Amir Globerson, Andreea Gane, the members of the MIT NLP group and the ACL reviewers for their suggestions and comments. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors, and do not necessarily reflect the views of the funding organizations.

References

Miguel Ballesteros and Joakim Nivre. 2012. MaltOptimizer: An optimization tool for MaltParser. In *EACL*. The Association for Computer Linguistics.

Miguel Ballesteros. 2013. Effective morphological feature selection with MaltOptimizer at the SPMRL 2013 shared task. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*. Association for Computational Linguistics.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06. Association for Computational Linguistics.

Venkat Chandrasekaran, Sujay Sanghavi, Pablo A Parrilo, and Alan S Willsky. 2011. Rank-sparsity incoherence for matrix decomposition. *SIAM Journal on Optimization*.

Volkan Cirik and Hüsnü Şensoy. 2013. The AI-KU system at the SPMRL 2013 shared task : Unsupervised features for dependency parsing. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*. Association for Computational Linguistics.

Shay B Cohen, Karl Stratos, Michael Collins, Dean P Foster, and Lyle Ungar. 2012. Spectral learning of latent-variable PCFGs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02. Association for Computational Linguistics.

R. Collobert and J. Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *International Conference on Machine Learning, ICML*.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *The Journal of Machine Learning Research*.

Tim Van de Cruys, Thierry Poibeau, and Anna Korhonen. 2013. A tensor-based factorization model of semantic compositionality. In *HLT-NAACL*. The Association for Computational Linguistics.

Paramveer S. Dhillon, Dean Foster, and Lyle Ungar. 2011. Multiview learning of word embeddings via CCA. In *Advances in Neural Information Processing Systems*.

A Evgeniou and Massimiliano Pontil. 2007. Multi-task feature learning. In *Advances in neural information processing systems: Proceedings of the 2006 conference*. The MIT Press.

Amir Globerson, Gal Chechik, Fernando Pereira, and Naftali Tishby. 2007. Euclidean embedding of co-occurrence data. *Journal of Machine Learning Research*.

Christopher Hillar and Lek-Heng Lim. 2009. Most tensor problems are NP-hard. *arXiv preprint arXiv:0911.1393*.

Daniel Hsu and Sham M Kakade. 2013. Learning mixtures of spherical gaussians: moment methods and spectral decompositions. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*. ACM.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10. Association for Computational Linguistics.

Terry Koo, Alexander M Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Angeliki Lazaridou, Eva Maria Vecchi, and Marco Baroni. 2013. Fish transporters and miracle homes: How compositional distributional semantics can help NP parsing. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Daniel D Lee and H Sebastian Seung. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature*.

Yariv Maron, Michael Lamar, and Elie Bienenstock. 2010. Sphere embedding: An application to part-of-speech induction. In *Advances in Neural Information Processing Systems*.

André FT Martins, Noah A Smith, Eric P Xing, Pedro MQ Aguiar, and Mário AT Figueiredo. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

André F. T. Martins, Noah A. Smith, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. 2011a. Dual decomposition with many overlapping components. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11. Association for Computational Linguistics.

André FT Martins, Noah A Smith, Pedro MQ Aguiar, and Mário AT Figueiredo. 2011b. Structured sparsity in structured prediction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

André FT Martins, Miguel B Almeida, and Noah A Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Yuval Marton, Nizar Habash, and Owen Rambow. 2010. Improving arabic dependency parsing with lexical and inflectional morphological features. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, SPMRL '10. Association for Computational Linguistics.

Yuval Marton, Nizar Habash, and Owen Rambow. 2011. Improving arabic dependency parsing with form-based and functional morphological features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics.

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

Peter Nilsson and Pierre Nugues. 2010. Automatic discovery of feature sets for dependency parsing. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. Coling 2010 Organizing Committee.

Joakim Nivre, Johan Hall, Jens Nilsson, Gülşen Eryiit, and Svetoslav Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*.

Alexander Rush and Slav Petrov. 2012a. Vine pruning for efficient multi-pass dependency parsing. In *The 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL '12)*.

Alexander M Rush and Slav Petrov. 2012b. Vine pruning for efficient multi-pass dependency parsing. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics*.

Nathan Srebro, Tommi Jaakkola, et al. 2003. Weighted low-rank approximations. In *ICML*.

Nathan Srebro, Jason Rennie, and Tommi S Jaakkola. 2004. Maximum-margin matrix factorization. In *Advances in neural information processing systems*.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, CoNLL '08. Association for Computational Linguistics.

Min Tao and Xiaoming Yuan. 2011. Recovering low-rank and sparse components of matrices from incomplete and noisy observations. *SIAM Journal on Optimization*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10. Association for Computational Linguistics.

Andrew E Waters, Aswin C Sankaranarayanan, and Richard Baraniuk. 2011. SpaRCS: Recovering low-rank and sparse matrices from compressive measurements. In *Advances in Neural Information Processing Systems*.

Hao Zhang and Ryan McDonald. 2012a. Generalized higher-order dependency parsing with cube pruning. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12. Association for Computational Linguistics.

Hao Zhang and Ryan McDonald. 2012b. Generalized higher-order dependency parsing with cube pruning. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics.

Hao Zhang, Liang Huang Kai Zhao, and Ryan McDonald. 2013. Online learning for inexact hypergraph search. In *Proceedings of EMNLP*.

Yuan Zhang, Tao Lei, Regina Barzilay, Tommi Jaakkola, and Amir Globerson. 2014. Steps to excellence: Simple inference with refined scoring of dependency trees. In *Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Tianyi Zhou and Dacheng Tao. 2011. Godec: Randomized low-rank & sparse matrix decomposition in noisy case. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*.

CoSimRank: A Flexible & Efficient Graph-Theoretic Similarity Measure

Sascha Rothe and Hinrich Schütze
Center for Information & Language Processing
University of Munich
sascha@cis.lmu.de

Abstract

We present *CoSimRank*, a graph-theoretic similarity measure that is efficient because it can compute a single node similarity without having to compute the similarities of the entire graph. We present equivalent formalizations that show CoSimRank's close relationship to Personalized PageRank and SimRank and also show how we can take advantage of fast matrix multiplication algorithms to compute CoSimRank. Another advantage of CoSimRank is that it can be flexibly extended from basic node-node similarity to several other graph-theoretic similarity measures. In an experimental evaluation on the tasks of synonym extraction and bilingual lexicon extraction, CoSimRank is faster or more accurate than previous approaches.

1 Introduction

Graph-theoretic algorithms have been successfully applied to many problems in NLP (Mihalcea and Radev, 2011). These algorithms are often based on PageRank (Brin and Page, 1998) and other centrality measures (e.g., (Erkan and Radev, 2004)). An alternative for tasks involving similarity is SimRank (Jeh and Widom, 2002). SimRank is based on the simple intuition that nodes in a graph should be considered as similar to the extent that their neighbors are similar. Unfortunately, SimRank has time complexity $\mathcal{O}(n^3)$ (where n is the number of nodes in the graph) and therefore does not scale to the large graphs that are typical of NLP.

This paper introduces CoSimRank,[1] a new graph-theoretic algorithm for computing node similarity that combines features of SimRank and PageRank. Our key observation is that to compute the similarity of two nodes, we need not consider all other nodes in the graph as SimRank does; instead, CoSimRank starts random walks from the two nodes and computes their similarity at each time step. This offers large savings in computation time if we only need the similarities of a small subset of all n^2 node similarities.

These two cases – computing a few similarities and computing many similarities – correspond to two different representations we can compute CoSimRank on: a vector representation, which is fast for only a few similarities, and a matrix representation, which can take advantage of fast matrix multiplication algorithms.

CoSimRank can be used to compute many variations of basic node similarity – including similarity for graphs with weighted and typed edges and similarity for sets of nodes. Thus, CoSimRank has the added advantage of being a flexible tool for different types of applications.

The extension of CoSimRank to *similarity across graphs* is important for the application of bilingual lexicon extraction: given a set of correspondences between nodes in two graphs A and B (corresponding to two different languages), a pair of nodes $(a \in A, b \in B)$ is a good candidate for a translation pair if their node similarity is high. In an experimental evaluation, we show that CoSimRank is more efficient and more accurate than both SimRank and PageRank-based algorithms.

This paper is structured as follows. Section 2 discusses related work. Section 3 introduces CoSimRank. In Section 4, we compare CoSimRank and SimRank. By providing some useful extensions, we demonstrate the great flexibility of CoSimRank (Section 5). We perform an experimental evaluation of CoSimRank in Section 6. Section 7 summarizes the paper.

2 Related Work

Our work is unsupervised. We therefore do not review graph-based methods that make extensive

[1] Code available at code.google.com/p/cistern

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1392–1402,
Baltimore, Maryland, USA, June 23-25 2014.

use of supervised learning (e.g., de Melo and Weikum (2012)).

Since the original version of *SimRank* (Jeh and Widom, 2002) has complexity $\mathcal{O}(n^4)$, many extensions have been proposed to speed up its calculation. A Monte Carlo algorithm, which is scalable to the whole web, was suggested by Fogaras and Rácz (2005). However, in an evaluation of this algorithm we found that it does not give competitive results (see Section 6). A matrix representation of SimRank called *SimFusion* (Xi et al., 2005) improves the computational complexity from $\mathcal{O}(n^4)$ to $\mathcal{O}(n^3)$. Lizorkin et al. (2010) also reduce complexity to $\mathcal{O}(n^3)$ by selecting essential node pairs and using partial sums. They also give a useful overview of SimRank, SimFusion and the Monte Carlo methods of Fogaras and Rácz (2005). A non-iterative computation for SimRank was introduced by Li et al. (2010). This is especially useful for dynamic graphs. However, all of these methods have to run SimRank on the entire graph and are not efficient enough for very large graphs. We are interested in applications that only need a fraction of all $\mathcal{O}(n^2)$ pairwise similarities. The algorithm we propose below is an order of magnitude faster in such applications because it is based on a local formulation of the similarity measure.[2]

Apart from SimRank, many other similarity measures have been proposed. Leicht et al. (2006) introduce a similarity measure that is also based on the idea that nodes are similar when their neighbors are, but that is designed for bipartite graphs. However, most graphs in NLP are not bipartite and Jeh and Widom (2002) also proposed a SimRank variant for bipartite graphs.

Another important similarity measure is cosine similarity of *Personalized PageRank* (PPR) vectors. We will refer to this measure as *PPR+cos*. Hughes and Ramage (2007) find that PPR+cos has high correlation with human similarity judgments on WordNet-based graphs. Agirre et al. (2009) use PPR+cos for WordNet and for cross-lingual studies. Like CoSimRank, PPR+cos is efficient when computing single node pair similarities; we therefore use it as one of our baselines below. This method is also used by Chang et al. (2013) for semantic relatedness. They also experimented with Euclidean distance and KL-divergence. Interestingly, a simpler method performed best when comparing with human similarity judgments. In this method only the entries corresponding to the compared nodes are used for a similarity score. Rao et al. (2008) compared PPR+cos to other graph based similarity measures like shortest-path and bounded-length random walks. PPR+cos performed best except for a new similarity measure based on commute time. We do not compare against this new measure as it uses the graph Laplacian and so cannot be computed for a single node pair.

One reason CoSimRank is efficient is that we need only compute a few iterations of the random walk. This is often true of this type of algorithm; cf. (Schütze and Walsh, 2008).

LexRank (Erkan and Radev, 2004) is similar to PPR+cos in that it combines PageRank and cosine; it initializes the sentence similarity matrix of a document using cosine and then applies PageRank to compute lexical centrality. Despite this superficial relatedness, applications like lexicon extraction that look for *similar entities* and applications that look for *central entities* are quite different.

In addition to faster versions of SimRank, there has also been work on extensions of SimRank. Dorow et al. (2009) and Laws et al. (2010) extend SimRank to edge weights, edge labels and multiple graphs. We use their Multi-Edge Extraction (MEE) algorithm as one of our baselines below. A similar graph of dependency structures was built by Minkov and Cohen (2008). They applied different similarity measures, e.g., cosine of dependency vectors or a new algorithm called *path-constrained* graph walk, on synonym extraction (Minkov and Cohen, 2012). We compare CoSimRank with their results in our experiments (see Section 6).

Some other applications of SimRank or other graph based similarity measures in NLP include work on document similarity (Li et al., 2009), the transfer of sentiment information between languages (Scheible et al., 2010) and named entity disambiguation (Han and Zhao, 2010). Hoang and Kan (2010) use SimRank for related work summarization. Muthukrishnan et al. (2010) combine link based similarity and content based similarity for document clustering and classification.

These approaches use at least one of cosine similarity, PageRank and SimRank. CoSimRank can either be interpreted as an efficient version of Sim-

[2] A reviewer suggests that CoSimRank is an efficient version of SimRank in a way analogous to SALSA's (Lempel and Moran, 2000) relationship to HITS (Kleinberg, 1999) in that different aspects of similarity are decoupled.

Rank or as a version of Personalized PageRank for similarity measurement. The novelty is that we compute similarity for vectors that are induced using a new algorithm, so that the similarity measurement is much more efficient when an application only needs a fraction of all $\mathcal{O}(n^2)$ pairwise similarities.

3 CoSimRank

We first first give an intuitive introduction of CoSimRank as a Personalized PageRank (PPR) derivative. Later on, we will give a matrix formulation to compare CoSimRank with SimRank.

3.1 Personalized PageRank

Haveliwala (2002) introduced Personalized PageRank – or topic-sensitive PageRank – based on the idea that the uniform damping vector $p^{(0)}$ can be replaced by a personalized vector, which depends on node i. We usually set $p^{(0)}(i) = e_i$, with e_i being a vector of the standard basis, i.e., the i^{th} entry is 1 and all other entries are 0. The PPR vector of node i is given by:

$$p^{(k)}(i) = dAp^{(k-1)}(i) + (1-d)p^{(0)}(i) \quad (1)$$

where A is the stochastic matrix of the Markov chain, i.e., the row normalized adjacency matrix. The damping factor $d \in (0,1)$ ensures that the computation converges. The PPR vector after k iterations is given by $p^{(k)}$.

To visualize this formula, one can imagine a random surfer starting at node i and following one of the links with probability d or jumping back to the starting node i with probability $(1-d)$. Entry i of the converged PPR vector represents the probability that the random surfer is on node i after an unlimited number of steps.

To simulate the behavior of SimRank we will simplify this equation and set the damping factor $d = 1$. We will re-add a damping factor later in the calculation.

$$p^{(k)} = Ap^{(k-1)} \quad (2)$$

Note that the personalization vector $p^{(0)}$ was eliminated, but is still present as the starting vector of the iteration.

3.2 Similarity of vectors

Let $p(i)$ be the PPR vector of node i. The cosine of two vectors u and v is computed by dividing the inner product $\langle u, v\rangle$ by the lengths of the vectors. The cosine of two PPR vectors can be used as a similarity measure for the corresponding nodes (Hughes and Ramage, 2007; Agirre et al., 2009):

$$s(i,j) = \frac{\langle p(i), p(j)\rangle}{|p(i)||p(j)|} \quad (3)$$

This measure $s(i,j)$ looks at the probability that a random walker is on a certain edge after an *unlimited number of steps*. This is potentially problematic as the example in Figure 1 shows. The PPR vectors of *suit* and *dress* will have some weight on *tailor*, which is good. However, the PPR vector of *law* will also have a non-zero weight for *tailor*. So *law* and *dress* are similar because of the node *tailor*. This is undesirable.

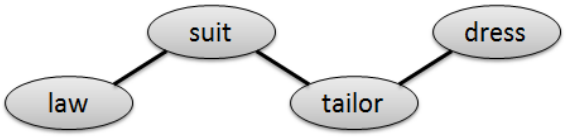


Figure 1: Graph motivating CoSimRank algorithm. Whereas PPR gives relatively high similarity to the pair (law,suit), CoSimRank assigns the pair similarity 0.

We can prevent this type of spurious similarity by *taking into account the path the random surfer took to get to a particular node*. We formalize this by defining CoSimRank $s(i,j)$ as follows:

$$s(i,j) = \sum_{k=0}^{\infty} c^k \langle p^{(k)}(i), p^{(k)}(j)\rangle \quad (4)$$

where $p^{(k)}(i)$ is the PPR vector of node i from Eq. 2 after k iterations. We compare the PPR vectors at each time step k. The sum of all similarities is the value of CoSimRank, i.e., the final similarity. We add a damping factor c, so that early meetings are more valuable than later meetings.

To compute the similarity of two vectors u and v we use the inner product $\langle\cdot,\cdot\rangle$ in Eq. 4 for two reasons:

1. This is similar to cosine similarity except that the 1-norm is used instead of the 2-norm. Since our vectors are probability vectors, we have

$$\frac{\langle p(i), p(j)\rangle}{|p(i)||p(j)|} = \langle p(i), p(j)\rangle$$

for the 1-norm.[3]

2. Without expensive normalization, we can give a simple matrix formalization of CoSimRank and compute it efficiently using fast matrix multiplication algorithms.

Later on, the following iterative computation of CoSimRank will prove useful:

$$s^{(k)}(i,j) = c^k \langle p^{(k)}(i), p^{(k)}(j) \rangle + s^{(k-1)}(i,j) \tag{5}$$

3.3 Matrix formulation

The matrix formulation of CoSimRank is:

$$\begin{aligned} S^{(0)} &= E \\ S^{(1)} &= cAA^T + S^{(0)} \\ S^{(2)} &= c^2 A^2 (A^T)^2 + S^{(1)} \\ &\dots \\ S^{(k)} &= c^k A^k (A^T)^k + S^{(k-1)} \end{aligned} \tag{6}$$

We will see in Section 5 that this formulation is the basis for a very efficient version of CoSimRank.

3.4 Convergence properties

As the PPR vectors have only positive values, we can easily see in Eq. 4 that the CoSimRank of one node pair is monotonically non-decreasing. For the dot product of two vectors, the Cauchy-Schwarz inequality gives the upper bound:

$$\langle u, v \rangle \leq \|u\| \, \|v\|$$

where $\|x\|$ is the norm of x. From Eq. 2 we get $\left\|p^{(k)}\right\|_1 = 1$, where $\|\cdot\|_1$ is the 1-norm. We also know from elementary functional analysis that the 1-norm is the biggest of all p-norms and so one has $\left\|p^{(k)}\right\| \leq 1$. It follows that CoSimRank grows more slowly than a geometric series and converges if $|c| < 1$:

$$s(i,j) \leq \sum_{k=0}^{\infty} c^k = \frac{1}{1-c}$$

If an upper bound of 1 is desired for $s(i,j)$ (instead of $1/(1-c)$), then we can use s':

$$s'(i,j) = (1-c)s(i,j)$$

[3]This type of similarity measure has also been used and investigated by Ó Séaghdha and Copestake (2008), Cha (2007), Jebara et al. (2004) (probability product kernel) and (Jaakkola et al., 1999) (Fisher kernel) among others.

4 Comparison to SimRank

The original SimRank equation can be written as follows (Jeh and Widom, 2002):

$$r(i,j) = \begin{cases} 1, & \text{if } i = j \\ \frac{c}{|N(i)||N(j)|} \sum\limits_{\substack{k \in N(i) \\ l \in N(j)}} r(k,l), & \text{else} \end{cases}$$

where $N(i)$ denotes the nodes connected to i. SimRank is computed iteratively. With A being the normalized adjacency matrix we can write SimRank in matrix formulation:

$$\begin{aligned} R^{(0)} &= E \\ R^{(k)} &= \max\{cAR^{(k-1)}A^T, R^{(0)}\} \end{aligned} \tag{7}$$

where the maximum of two matrices refers to the element-wise maximum. We will now prove by induction that the matrix formulation of CoSimRank (Eq. 6) is equivalent to:

$$S'^{(k)} = cAS'^{(k-1)}A^T + S^{(0)} \tag{8}$$

and thus very similar to SimRank (Eq. 7).

The base case $S^{(1)} = S'^{(1)}$ is trivial. Inductive step:

$$\begin{aligned} S'^{(k)} &\overset{(8)}{=} cAS'^{(k-1)}A^T + S^{(0)} \\ &= cA(c^{k-1}A^{k-1}(A^T)^{k-1} + S^{(k-2)})A^T + S^{(0)} \\ &= c^k A^k (A^T)^k + cAS^{(k-2)}A^T + S^{(0)} \\ &= c^k A^k (A^T)^k + S^{(k-1)} \overset{(6)}{=} S^{(k)} \end{aligned}$$ □

Comparing Eqs. 7 and 8, we see that SimRank and CoSimRank are very similar except that they initialize the similarities on the diagonal differently. Whereas SimRank sets each of these entries back to one at each iteration, CoSimRank adds one. Thus, when computing the two similarity measures iteratively, the diagonal element (i,i) will be set to 1 by both methods for those initial iterations for which this entry is 0 for $cAS^{(k-1)}A^T$ (i.e., before applying either max or add). The methods diverge when the entry is $\neq 0$ for the first time.

Complexity of computing all n^2 similarities. The matrix formulas of both SimRank (Eq. 7) and CoSimRank (Eq. 8) have time complexity $\mathcal{O}(n^3)$ or – if we want to take the higher efficiency of computation for sparse graphs into account – $\mathcal{O}(dn^2)$ where n is the number of nodes and d the

average degree. Space complexity is $\mathcal{O}(n^2)$ for both algorithms.

Complexity of computing $k^2 \ll n^2$ similarities. In most cases, we only want to compute k^2 similarities for k nodes. For CoSimRank, we compute the k PPR vectors in $\mathcal{O}(kdn)$ (Eq. 2) and compute the k^2 similarities in $\mathcal{O}(k^2n)$ (Eq. 5). If $d < k$, then the time complexity of CoSimRank is $\mathcal{O}(k^2n)$. If we only compute a single similarity, then the complexity is $\mathcal{O}(dn)$. In contrast, the complexity of SimRank is the same as in the all-similarities case: $\mathcal{O}(dn^2)$. It is not obvious how to design a lower-complexity version of SimRank for this case. Thus, we have reduced SimRank's cubic time complexity to a quadratic time complexity for CoSimRank or – assuming that the average degree d does not depend on n – SimRank's quadratic time complexity to linear time complexity for the case of computing few similarities.

Space complexity for computing k^2 similarities is $\mathcal{O}(kn)$ since we need only store k vectors, not the complete similarity matrix. This complexity can be exploited even for the all similarities application: If the matrix formulation cannot be used because the $\mathcal{O}(n^2)$ similarity matrix is too big for available memory, then we can compute all similarities in batches – and if desired in parallel – whose size is chosen such that the vectors of each batch still fit in memory.

In summary, CoSimRank and SimRank have similar space and time complexities for computing all n^2 similarities. For the more typical case that we only want to compute a fraction of all similarities, we have recast the global SimRank formulation as a local CoSimRank formulation. As a result, time and space complexities are much improved. In Section 6, we will show that this is also true in practice.

5 Extensions

We will show now that the basic CoSimRank algorithm can be extended in a number of ways and is thus a flexible tool for different NLP applications.

5.1 Weighted edges

The use of weighted edges was first proposed in the PageRank patent. It is straightforward and easy to implement by replacing the row normalized adjacency matrix A with an arbitrary stochastic matrix P. We can use this edge weighted PageRank for CoSimRank.

5.2 CoSimRank across graphs

We often want to compute the similarity of nodes in *two different graphs* with a known node-node correspondence; this is the scenario we are faced with in the lexicon extraction task (see Section 6). A variant of SimRank for this task was presented by Dorow et al. (2009). We will now present an equivalent method for CoSimRank. We denote the number of nodes in the two graphs U and V by $|U|$ and $|V|$, respectively. We compute PPR vectors $p \in \mathbb{R}^{|U|}$ and $q \in \mathbb{R}^{|V|}$ for each graph. Let $S^{(0)} \in \mathbb{R}^{|U| \times |V|}$ be the known node-node correspondences. The analog of CoSimRank (Eq. 4) for two graphs is then:

$$s(i,j) = \sum_{k=0}^{\infty} c^k \sum_{(u,v) \in S^{(0)}} p_u^{(k)}(i) q_v^{(k)}(j) \quad (9)$$

The matrix formulation (cf. Eq. 6) is:

$$S^{(k)} = c^k A^k S^{(0)} (B^T)^k + S^{(k-1)} \quad (10)$$

where A and B are row-normalized adjacency matrices. We can interpret $S^{(0)}$ as a change of basis. A similar approach for word embeddings was published by Mikolov et al. (2013). They call $S^{(0)}$ the translation matrix.

5.3 Typed edges

To be able to directly compare to prior work in our experiments, we also present a method to integrate a set of typed edges $\mathcal{T}$ in the CoSimRank calculation. For this we will compute a similarity matrix for each edge type τ and merge them into one matrix for the next iteration:

$$S^{(k)} = \left(\frac{c}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} A_\tau S^{(k-1)} B_\tau^T \right) + S^{(0)} \quad (11)$$

This formula is identical to the random surfer model where two surfers only meet iff they are on the same node and used the same edge type to get there. A more strict claim would be to use the same edge type at any time of their journey:

$$S^{(k)} = \frac{c^k}{|\mathcal{T}|^k} \sum_{\tau \in \mathcal{T}^k} \left(\prod_{i=1}^{k} A_{\tau_i} \right) S^{(0)} \left(\prod_{i=0}^{k-1} B_{\tau_{k-i}}^T \right) + S^{(k-1)} \quad (12)$$

We will not use Eq. 12 due to its space complexity.

5.4 Similarity of sets of nodes

CoSimRank can also be used to compute the similarity $s(V, W)$ of two sets V and W of nodes, e.g., short text snippets. We are not including this method in our experiments, but we will give the equation here, as traditional document similarity measures (e.g., cosine similarity) perform poorly on this task although there also are known alternatives with good results (Sahami and Heilman, 2006). For a set V, the initial PPR vector is given by:

$$p_i^{(0)}(V) = \begin{cases} \frac{1}{|V|}, & \text{if } i \in V \\ 0, & \text{else} \end{cases}$$

We then reuse Eq. 4 to compute $s(V, W)$:

$$s(V, W) = \sum_{k=0}^{\infty} c^k \langle p^{(k)}(V), p^{(k)}(W) \rangle$$

In summary, modifications proposed for SimRank (weighted and typed edges, similarity across graphs) as well as modifications proposed for PageRank (sets of nodes) can also be applied to CoSimRank. This makes CoSimRank a very flexible similarity measure.

We will test the first three extensions experimentally in the next section and leave similarity of node sets for future work.

6 Experiments

We evaluate CoSimRank for the tasks of synonym extraction and bilingual lexicon extraction. We use the basic version of CoSimRank (Eq. 4) for synonym extraction and the two-graph version (Eq. 9) for lexicon extraction, both with weighted edges. Our motivation for this application is that two words that are synonyms of each other should have similar lexical neighbors and that two words that are translations of each other should have neighbors that correspond to each other; thus, in each case the nodes should be similar in the graph-theoretic sense and CoSimRank should be able to identify this similarity.

We use the English and German graphs published by Laws et al. (2010), including edge weighting and normalization. Nodes are nouns, adjectives and verbs occurring in Wikipedia. There are three types of edges, corresponding to three types of syntactic configurations extracted from the parsed Wikipedias: adjective-noun, verb-object and noun-noun coordination. Table 1 gives examples and number of nodes and edges.

Edge types

relation	entities	description	example
amod	a, v	adjective-noun	a fast car
dobj	v, n	verb-object	drive a car
ncrd	n, n	noun-noun	cars and busses

Graph statistics

nodes		nouns	adjectives	verbs
	de	34,544	10,067	2,828
	en	22,258	12,878	4,866
edges		ncrd	amod	dobj
	de	65,299	417,151	143,905
	en	288,878	686,069	510,351

Table 1: Edge types (above) and number of nodes and edges (below)

6.1 Baselines

We propose CoSimRank as an efficient algorithm for computing the similarity of nodes in a graph. Consequently, we compare against the two main methods for this task in NLP: SimRank and extensions of PageRank.

We also compare against the MEE (Multi-Edge Extraction) variant of SimRank (Dorow et al., 2009), which handles labeled edges more efficiently than SimRank:

$$S'^{(k)} = \frac{c}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} A_\tau S^{(k-1)} B_\tau^T$$

$$S^{(k)} = \max\{S'^{(k)}, S^{(0)}\}$$

where A_τ is the row-normalized adjacency matrix for edge type τ (see edge types in Table 1).

Apart from SimRank, extensions of PageRank are the main methods for computing the similarity of nodes in graphs in NLP (e.g., Hughes and Ramage (2007), Agirre et al. (2009) and other papers discussed in related work). Generally, these methods compute the Personalized PageRank for each node (see Eq. 1). When the computation has converged, the similarity of two nodes is given by the cosine similarity of the Personalized PageRank vectors. We implemented this method for our experiments and call it PPR+cos.

6.2 Synonym Extraction

We use *TS68*, a test set of 68 synonym pairs published by Minkov and Cohen (2012) for evaluation. This gold standard lists a single word as the

	P@1	P@10	MRR
one-synonym			
PPR+cos	20.6%	52.9%	0.32
SimRank	**25.0**%	61.8%	**0.37**
CoSimRank	**25.0**%	61.8%	**0.37**
Typed CoSimRank	23.5%	**63.2**%	**0.37**
extended			
PPR+cos	32.6%	73.5%	0.48
SimRank	**45.6**%	**83.8**%	**0.59**
CoSimRank	**45.6**%	**83.8**%	**0.59**
Typed CoSimRank	44.1%	**83.8**%	**0.59**

Table 2: Results for synonym extraction on TS68. Best result in each column in bold.

correct synonym even if there are several equally acceptable near-synonyms (see Table 3 for examples). We call this the one-synonym evaluation. Three native English speakers were asked to mark synonyms, that were proposed by a baseline or by CoSimRank, i.e. ranked in the top 10. If all three of them agreed on one word as being a synonym in at least one meaning, we added this as a correct answer to the test set. We call this the "extended" evaluation (see Table 2).

Synonym extraction is run on the English graph. To calculate PPR+cos, we computed 20 iterations with a decay factor of 0.8 and used the cosine similarity with the 2-norm in the denominator to compare two vectors. For the other three methods, we also used a decay factor of 0.8 and computed 5 iterations. Recall that CoSimRank uses the simple inner product $\langle \cdot, \cdot \rangle$ to compare vectors.

Our evaluation measures are proportion of words correctly translated by word in the top position (P@1), proportion of words correctly translated by a word in one of the top 10 positions (P@10) and Mean Reciprocal Rank (MRR). CoSimRank's MRR scores of 0.37 (one-synonym) and 0.59 (extended) are the same or better than all baselines (see Table 2). CoSimRank and SimRank have the same P@1 and P@10 accuracy (although they differed on some decisions). CoSimRank is better than PPR+cos on both evaluations, but as this test set is very small, the results are not significant. Table 3 shows a sample of synonyms proposed by CoSimRank.

Minkov and Cohen (2012) tested cosine and random-walk measures on grammatical relationships (similar to our setup) as well as on cooccurrence statistics. The MRR scores for these methods range from 0.29 to 0.59. (MRR is equivalent to MAP as reported by Minkov and Cohen (2012) when there is only one correct answer.) Their best number (0.59) is better than our one-synonym result; however, they performed manual postprocessing of results – e.g., discarding words that are morphologically or semantically related to other words in the list – so our fully automatic results cannot be directly compared.

keyword	expected	extracted
movie	film	**film**
modern	contemporary	**contemporary**
demonstrate	protest	**show**
attractive	appealing	beautiful
economic	profitable	financial
close	shut	open

Table 3: Examples for extracted synonyms. Correct synonyms according to extended evaluation in bold.

6.3 Lexicon Extraction

We evaluate lexicon extraction on TS1000, a test set of 1000 items, (Laws et al., 2010) each consisting of an English word and its German translations. For lexicon extraction, we use the same parameters as in the synonym extraction task for all four similarity measures. We use a seed dictionary of 12,630 word pairs to establish node-node correspondences between the two graphs. We remove a search keyword from the seed dictionary before calculating similarities for it, something that the architecture of CoSimRank makes easy because we can use a different seed dictionary $S^{(0)}$ for every keyword.

Both CoSimRank methods outperform SimRank significantly (see Table 4). The difference between CoSimRank with and without typed edges is not significant. (This observation was also made for SimRank on a smaller graph and test set (Laws et al., 2010).)

PPR+cos's performance at 14.8% correct translations is much lower than SimRank and CoSimRank. The disadvantage of this similarity measure is significant and even more visible on bilingual lexicon extraction than on synonym extraction (see Table 2). The reason might be that we are not comparing the whole PPR vector anymore,

	P@1	P@10
PPR+cos	14.8%†	45.7%†
SimRank MEE	48.0%†	76.0%†
CoSimRank	61.1%	**84.0%**
Typed CoSimRank	**61.4%**	83.9%

Table 4: Results for bilingual lexicon extraction (TS1000 EN → DE). Best result in each column in bold.

	synonym extraction (68 word pairs)	lexicon extraction (1000 word pairs)
PPR+cos	2,228	**2,195**
SimRank	23,423	14,418
CoSimRank	**524**	2,342
Typed CoSimRank	615	6,108

Table 5: Execution times in minutes for CoSimRank and the baselines. Best result in each column in bold.

but only entries which occur in the seed dictionary (see Eq. 9). As the seed dictionary contains 12,630 word pairs, this means that only every fourth entry of the PPR vector (the German graph has 47,439 nodes) is used for similarity calculation. This is also true for CoSimRank, but it seems that CoSimRank is more stable because we compare more than one vector.

We also experimented with the method of Fogaras and Rácz (2005). We tried a number of different ways of modifying it for weighted graphs: (i) running the random walks with the weighted adjacency matrix as Markov matrix, (ii) storing the weight (product of each edge weight) of a random walk and using it as a factor if two walks meet and (iii) a combination of both. We needed about 10,000 random walks in all three conditions. As a result, the computational time was approximately 30 minutes per test word, so this method is even slower than SimRank for our application. The accuracies P@1 and P@10 were worse in all experiments than those of CoSimRank.

6.4 Run time performance

Table 5 compares the run time performance of CoSimRank with the baselines. We ran all experiments on a 64-bit Linux machine with 64 Intel Xenon X7560 2.27Ghz CPUs and 1TB RAM. The calculated time is the sum of the time spent in user mode and the time spent in kernel mode. The actual wall clock time was significantly lower as we used up to 64 CPUs.

Compared to SimRank, CoSimRank is more than 40 times faster on synonym extraction and six times faster on lexicon extraction. SimRank is at a disadvantage because it computes all similarities in the graph regardless of the size of the test set; it is particularly inefficient on synonym extraction because the English graph contains a large number of edges (see Table 1).

Compared to PPR+cos, CoSimRank is roughly four times faster on synonym extraction and has comparable performance on lexicon extraction. We compute 20 iterations of PPR+cos to reach convergence and then calculate a single cosine similarity. For CoSimRank, we need only compute five iterations to reach convergence, but we have to compute a vector similarity in each iteration. The counteracting effects of fewer iterations and more vector similarity computations can give either CoSimRank or PPR+cos an advantage, as is the case for synonym extraction and lexicon extraction, respectively.

CoSimRank should generally be three times faster than typed CoSimRank since the typed version has to repeat the computation for each of the three types. This effect is only visible on the larger test set (lexicon extraction) because the general computation overhead is about the same on a smaller test set.

6.5 Comparison with WINTIAN

Here we address inducing a bilingual lexicon from a seed set based on grammatical relations found by a parser. An alternative approach is to induce a bilingual lexicon from Wikipedia's interwiki links (Rapp et al., 2012). These two approaches have different strengths and weaknesses; e.g., the interwiki-link-based approach does not require a seed set, but it can only be applied to comparable corpora that consist of corresponding – although not necessarily "parallel" – documents.

Despite these differences it is still interesting to compare the two algorithms. Rapp et al. (2012) kindly provided their test set to us. It contains 1000 English words and a single correct German translation for each. We evaluate on a subset we call TS774 that consists of the 774 test word pairs that are in the intersection of words covered by the

†significantly worse than CoSimRank ($\alpha = 0.05$, one-tailed Z-Test)

	P@1	P@10
Wintian	**43.8%**	55.4%†
CoSimRank	43.0%	**73.6%**

Table 6: Results for bilingual lexicon extraction (TS774 DE → EN). Best result in each column in bold.

WINTIAN Wikipedia data (Rapp et al., 2012) and words covered by our data. Most of the 226 missing word pairs are adverbs, prepositions and plural forms that are not covered by our graphs due to the construction algorithm we use: lemmatization, restriction to adjectives, nouns and verbs etc.

Table 6 shows that CoSimRank is slightly, but not significantly worse than WINTIAN on P@1 (43.0 vs 43.8), but significantly better on P@10 (73.6 vs 55.4).[4] The reason could be that CoSimRank is a more effective algorithm than WINTIAN; but the different initializations (seed set vs interwiki links) or the different linguistic representations (grammatical relations vs bag-of-words) could also be responsible.

6.6 Error Analysis

The results on TS774 can be considered conservative since only one translation is accepted as being correct. In reality other translations might also be acceptable (e.g., both *street* and *road* for *Straße*). In contrast, TS1000 accepts more than one correct translation. Additionally, TS774 was created by translating English words into German (using Google translate). We are now testing the reverse direction. So we are doomed to fail if the original English word is a less common translation of an ambiguous German word. For example, the English word *gulf* was translated by Google to *Golf*, but the most common sense of *Golf* is the sport. Hence our algorithm will incorrectly translate it back to *golf*.

As we can see in Table 7, we also face the problems discussed by Laws et al. (2010): the algorithm sometimes picks cohyponyms (which can still be seen as reasonable) and antonyms (which are clear errors).

Contrary to our intuition, the edge-typed variant of CoSimRank did not perform significantly better than the non-edge-typed version. Looking at Table 1, we see that there is only one edge type connecting adjectives. The same is true for verbs. The random surfer only has a real choice between different edge types when she is on a noun node. Combined with the fact that only the last edge type is important this has absolutely no effect for a random surfer meeting at adjectives or verbs.

Two possible solutions would be (i) to use more fine-grained edge types, (ii) to apply Eq. 12, in which the edge type of each step is important. However, this will increase the memory needed for calculation.

[4] We achieved better results for CoSimRank by optimizing the damping factor, but in this paper, we only present results for a fixed damping factor of 0.8.

keyword	gold standard	CoSimRank
arm	poor	**impoverished**
erreichen	reach	**achieve**
gehen	go	**walk**
direkt	directly	**direct**
weit	far	further
breit	wide	narrow
reduzieren	reduce	increase
Stunde	hour	second
Westen	west	southwest
Junge	boy	child

Table 7: Examples for CoSimRank translation errors on TS774. We counted translations as incorrect if they were not listed in the gold standard even if they were correct translations according to *www.dict.cc* (in bold).

7 Summary

We have presented *CoSimRank*, a new similarity measure that can be computed for a single node pair without relying on the similarities in the whole graph. We gave two different formalizations of CoSimRank: (i) a derivation from Personalized PageRank and (ii) a matrix representation that can take advantage of fast matrix multiplication algorithms. We also presented extensions of CoSimRank for a number of applications, thus demonstrating the flexibility of CoSimRank as a similarity measure.

We showed that CoSimRank is superior to SimRank in time and space complexity; and we demonstrated that CoSimRank performs better than PPR+cos on two similarity computation tasks.

Acknowledgments. This work was supported by DFG (SCHU 2246/2-2).

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 19–27.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. In *WWW*, pages 107–117.

Sung-Hyuk Cha. 2007. Comprehensive survey on distance/similarity measures between probability density functions. *Mathematical Models and Methods in Applied Sciences*, 1(4):300–307.

Ching-Yun Chang, Stephen Clark, and Brian Harrington. 2013. Getting creative with semantic similarity. In *Semantic Computing (ICSC), 2013 IEEE Seventh International Conference on*, pages 330–333.

Gerard de Melo and Gerhard Weikum. 2012. Uwn: A large multilingual lexical knowledge base. In *ACL (System Demonstrations)*, pages 151–156.

Beate Dorow, Florian Laws, Lukas Michelbacher, Christian Scheible, and Jason Utt. 2009. A graph-theoretic algorithm for automatic extension of translation lexicons. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, GEMS '09, pages 91–95.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Intell. Res. (JAIR)*, 22:457–479.

Dániel Fogaras and Balázs Rácz. 2005. Scaling link-based similarity search. In *Proceedings of the 14th international conference on World Wide Web*, WWW '05, pages 641–650.

Xianpei Han and Jun Zhao. 2010. Structural semantic relatedness: a knowledge-based method to named entity disambiguation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 50–59.

Taher H. Haveliwala. 2002. Topic-sensitive pagerank. In *Proceedings of the 11th international conference on World Wide Web*, WWW '02, pages 517–526.

Cong Duy Vu Hoang and Min-Yen Kan. 2010. Towards automated related work summarization. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 427–435.

Thad Hughes and Daniel Ramage. 2007. Lexical semantic relatedness with random graph walks. In *EMNLP-CoNLL*, pages 581–589.

Tommi Jaakkola, David Haussler, et al. 1999. Exploiting generative models in discriminative classifiers. *Advances in neural information processing systems*, pages 487–493.

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

Glen Jeh and Jennifer Widom. 2002. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 538–543.

Jon M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632.

Florian Laws, Lukas ichelbacher, Beate Dorow, Christian Scheible, Ulrich Heid, and Hinrich Schütze. 2010. A linguistically grounded graph model for bilingual lexicon extraction. In *Coling 2010: Posters*, pages 614–622.

Elizabeth Leicht, Petter Holme, and Mark Newman. 2006. Vertex similarity in networks. *Physical Review E*, 73(2):026120.

Ronny Lempel and Shlomo Moran. 2000. The stochastic approach for link-structure analysis (salsa) and the tkc effect. *Computer Networks*, 33(1):387–401.

Pei Li, Zhixu Li, Hongyan Liu, Jun He, and Xiaoyong Du. 2009. Using link-based content analysis to measure document similarity effectively. In *Proceedings of the Joint International Conferences on Advances in Data and Web Management*, APWeb/WAIM '09, pages 455–467.

Cuiping Li, Jiawei Han, Guoming He, Xin Jin, Yizhou Sun, Yintao Yu, and Tianyi Wu. 2010. Fast computation of simrank for static and dynamic information networks. In *Proceedings of the 13th International Conference on Extending Database Technology*, EDBT '10, pages 465–476.

Dmitry Lizorkin, Pavel Velikhov, Maxim Grinev, and Denis Turdakov. 2010. Accuracy estimate and optimization techniques for simrank computation. *The VLDB Journal—The International Journal on Very Large Data Bases*, 19(1):45–66.

Rada Mihalcea and Dragomir Radev. 2011. *Graph-based natural language processing and information retrieval*. Cambridge University Press.

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

Einat Minkov and William W. Cohen. 2008. Learning graph walk based similarity measures for parsed text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 907–916.

Einat Minkov and William W. Cohen. 2012. Graph based similarity measures for synonym extraction from parsed text. In *Workshop Proceedings of TextGraphs-7 on Graph-based Methods for Natural Language Processing*, TextGraphs-7 '12, pages 20–24.

Pradeep Muthukrishnan, Dragomir Radev, and Qiaozhu Mei. 2010. Edge weight regularization over multiple graphs for similarity learning. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 374–383.

Diarmuid Ó Séaghdha and Ann Copestake. 2008. Semantic classification with distributional kernels. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 649–656.

Delip Rao, David Yarowsky, and Chris Callison-Burch. 2008. Affinity measures based on the graph Laplacian. In *Proceedings of the 3rd Textgraphs Workshop on Graph-Based Algorithms for Natural Language Processing*, TextGraphs-3, pages 41–48.

Reinhard Rapp, Serge Sharoff, and Bogdan Babych. 2012. Identifying word translations from comparable documents without a seed lexicon. In *LREC*, pages 460–466.

Mehran Sahami and Timothy D. Heilman. 2006. A web-based kernel function for measuring the similarity of short text snippets. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, pages 377–386.

Christian Scheible, Florian Laws, Lukas Michelbacher, and Hinrich Schütze. 2010. Sentiment translation through multi-edge graphs. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 1104–1112.

Hinrich Schütze and Michael Walsh. 2008. A graph-theoretic model of lexical syntactic acquisition. In *EMNLP*, pages 917–926.

Wensi Xi, Edward A. Fox, Weiguo Fan, Benyu Zhang, Zheng Chen, Jun Yan, and Dong Zhuang. 2005. Simfusion: measuring similarity using unified relationship matrix. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 130–137.

Is this a wampimuk? Cross-modal mapping between distributional semantics and the visual world

Angeliki Lazaridou and **Elia Bruni** and **Marco Baroni**
Center for Mind/Brain Sciences
University of Trento
{angeliki.lazaridou|elia.bruni|marco.baroni}@unitn.it

Abstract

Following up on recent work on establishing a mapping between vector-based semantic embeddings of words and the visual representations of the corresponding objects from natural images, we first present a simple approach to cross-modal vector-based semantics for the task of *zero-shot learning*, in which an image of a previously unseen object is mapped to a linguistic representation denoting its word. We then introduce *fast mapping*, a challenging and more cognitively plausible variant of the zero-shot task, in which the learner is exposed to new objects and the corresponding words in very limited linguistic contexts. By combining prior linguistic and visual knowledge acquired about words and their objects, as well as exploiting the limited new evidence available, the learner must learn to associate new objects with words. Our results on this task pave the way to realistic simulations of how children or robots could use existing knowledge to bootstrap grounded semantic knowledge about new concepts.

1 Introduction

Computational models of meaning that rely on corpus-extracted context vectors, such as LSA (Landauer and Dumais, 1997), HAL (Lund and Burgess, 1996), Topic Models (Griffiths et al., 2007) and more recent neural-network approaches (Collobert and Weston, 2008; Mikolov et al., 2013b) have successfully tackled a number of lexical semantics tasks, where context vector similarity highly correlates with various indices of semantic relatedness (Turney and Pantel, 2010). Given that these models are learned from naturally occurring data using simple associative techniques, various authors have advanced the claim that they might be also capturing some crucial aspects of how humans acquire and use language (Landauer and Dumais, 1997; Lenci, 2008).

However, the models induce the meaning of words entirely from their co-occurrence with other words, without links to the external world. This constitutes a serious blow to claims of cognitive plausibility in at least two respects. One is the *grounding problem* (Harnad, 1990; Searle, 1984). Irrespective of their relatively high performance on various semantic tasks, it is debatable whether models that have no access to visual and perceptual information can capture the holistic, grounded knowledge that humans have about concepts. However, a possibly even more serious pitfall of vector models is *lack of reference*: natural language is, fundamentally, a means to communicate, and thus our words must be able to *refer* to objects, properties and events in the outside world (Abbott, 2010). Current vector models are purely language-internal, solipsistic models of meaning. Consider the very simple scenario in which visual information is being provided to an agent about the current state of the world, and the agent's task is to determine the truth of a statement similar to *There is a dog in the room*. Although the agent is equipped with a powerful context vector model, this will not suffice to successfully complete the task. The model might suggest that the concepts of *dog* and *cat* are semantically related, but it has no means to determine the visual appearance of dogs, and consequently no way to verify the truth of such a simple statement.

Mapping words to the objects they denote is such a core function of language that humans are highly optimized for it, as shown by the so-called *fast mapping* phenomenon, whereby children can learn to associate a word to an object or property by a single exposure to it (Bloom, 2000; Carey, 1978; Carey and Bartlett, 1978; Heibeck and Markman, 1987). But lack of reference is not

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1403–1414,
Baltimore, Maryland, USA, June 23-25 2014.

only a theoretical weakness: Without the ability to refer to the outside world, context vectors are arguably useless for practical goals such as learning to execute natural language instructions (Branavan et al., 2009; Chen and Mooney, 2011), that could greatly benefit from the rich network of lexical meaning such vectors encode, in order to scale up to real-life challenges.

Very recently, a number of papers have exploited advances in automated feature extraction form images and videos to enrich context vectors with visual information (Bruni et al., 2014; Feng and Lapata, 2010; Leong and Mihalcea, 2011; Regneri et al., 2013; Silberer et al., 2013). This line of research tackles the grounding problem: Word representations are no longer limited to their linguistic contexts but also encode visual information present in images associated with the corresponding objects. In this paper, we rely on the same image analysis techniques but instead focus on the reference problem: We do not aim at enriching word representations with visual information, although this might be a side effect of our approach, but we address the issue of automatically mapping objects, as depicted in images, to the context vectors representing the corresponding words. This is achieved by means of a simple neural network trained to project image-extracted feature vectors to text-based vectors through a hidden layer that can be interpreted as a cross-modal semantic space.

We first test the effectiveness of our cross-modal semantic space on the so-called *zero-shot learning* task (Palatucci et al., 2009), which has recently been explored in the machine learning community (Frome et al., 2013; Socher et al., 2013). In this setting, we assume that our system possesses linguistic and visual information for a set of concepts in the form of text-based representations of words and image-based vectors of the corresponding objects, used for vision-to-language-mapping training. The system is then provided with visual information for a previously unseen object, and the task is to associate it with a word by cross-modal mapping. Our approach is competitive with respect to the recently proposed alternatives, while being overall simpler.

The aforementioned task is very demanding and interesting from an engineering point of view. However, from a cognitive angle, it relies on strong, unrealistic assumptions: The learner is asked to establish a link between a new object and a word for which they possess a full-fledged text-based vector extracted from a billion-word corpus. On the contrary, the first time a learner is exposed to a new object, the linguistic information available is likely also very limited. Thus, in order to consider vision-to-language mapping under more plausible conditions, similar to the ones that children or robots in a new environment are faced with, we next simulate a scenario akin to fast mapping. We show that the induced cross-modal semantic space is powerful enough that sensible guesses about the correct word denoting an object can be made, even when the linguistic context vector representing the word has been created from as little as 1 sentence containing it.

The contributions of this work are three-fold. First, we conduct experiments with simple image- and text-based vector representations and compare alternative methods to perform cross-modal mapping. Then, we complement recent work (Frome et al., 2013) and show that zero-shot learning scales to a large and noisy dataset. Finally, we provide preliminary evidence that cross-modal projections can be used effectively to simulate a fast mapping scenario, thus strengthening the claims of this approach as a full-fledged, fully inductive theory of meaning acquisition.

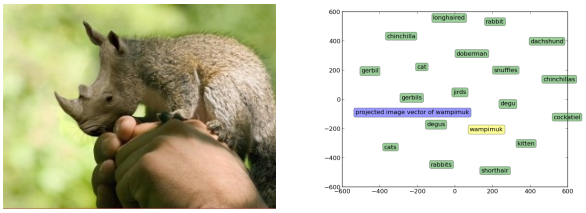
2 Related Work

The problem of establishing word reference has been extensively explored in computational simulations of cross-situational learning (see Fazly et al. (2010) for a recent proposal and extended review of previous work). This line of research has traditionally assumed artificial models of the external world, typically a set of linguistic or logical labels for objects, actions and possibly other aspects of a scene (Siskind, 1996). Recently, Yu and Siskind (2013) presented a system that induces word-object mappings from features extracted from short videos paired with sentences. Our work complements theirs in two ways. First, unlike Yu and Siskind (2013) who considered a limited lexicon of 15 items with only 4 nouns, we conduct experiments in a large search space containing a highly ambiguous set of potential target words for every object (see Section 4.1). Most importantly, by projecting visual representations of objects into a shared *semantic space*, we do not limit ourselves to establishing a link between ob-

jects and words. We induce a rich semantic representation of the multimodal concept, that can lead, among other things, to the discovery of important properties of an object even when we lack its linguistic label. Nevertheless, Yu and Siskind's system could in principle be used to initialize the vision-language mapping that we rely upon.

Closer to the spirit of our work are two very recent studies coming from the machine learning community. Socher et al. (2013) and Frome et al. (2013) focus on zero-shot learning in the vision-language domain by exploiting a shared visual-linguistic semantic space. Socher et al. (2013) learn to project unsupervised vector-based image representations onto a word-based semantic space using a neural network architecture. Unlike us, Socher and colleagues train an outlier detector to decide whether a test image should receive a known-word label by means of a standard supervised object classifier, or be assigned an unseen label by vision-to-language mapping. In our zero-shot experiments, we assume no access to an outlier detector, and thus, the search for the correct label is performed in the full concept space. Furthermore, Socher and colleagues present a much more constrained evaluation setup, where only 10 concepts are considered, compared to our experiments with hundreds or thousands of concepts.

Frome et al. (2013) use linear regression to transform vector-based image representations onto vectors representing the same concepts in linguistic semantic space. Unlike Socher et al. (2013) and the current study that adopt simple unsupervised techniques for constructing image representations, Frome et al. (2013) rely on a supervised state-of-the-art method: They feed low-level features to a deep neural network trained on a supervised object recognition task (Krizhevsky et al., 2012). Furthermore, their text-based vectors encode very rich information, such as $\vec{king} - \vec{man} + \vec{woman} = \vec{queen}$ (Mikolov et al., 2013c). A natural question we aim to answer is whether the success of cross-modal mapping is due to the high-quality embeddings or to the general algorithmic design. If the latter is the case, then these results could be extended to traditional distributional vectors bearing other desirable properties, such as high interpretability of dimensions.



(a) (b)

Figure 1: A potential *wampimuk* (a) together with its projection onto the linguistic space (b).

3 Zero-shot learning and fast mapping

"We found a cute, hairy *wampimuk* sleeping behind the tree." Even though the previous statement is certainly the first time one hears about *wampimuks*, the linguistic context already creates some visual expectations: Wampimuks probably resemble small animals (Figure 1a). This is the scenario of *zero-shot learning*. Moreover, if this is also the first linguistic encounter of that concept, then we refer to the task as *fast mapping*.

Concretely, we assume that concepts, denoted for convenience by word labels, are represented in linguistic terms by vectors in a text-based distributional semantic space (see Section 4.3). Objects corresponding to concepts are represented in visual terms by vectors in an image-based semantic space (Section 4.2). For a subset of concepts (e.g., a set of animals, a set of vehicles), we possess information related to both their linguistic and visual representations. During training, this cross-modal vocabulary is used to induce a projection function (Section 4.4), which – intuitively – represents a mapping between visual and linguistic dimensions. Thus, this function, given a visual vector, returns its corresponding linguistic representation. At test time, the system is presented with a previously unseen object (e.g., *wampimuk*). This object is projected onto the linguistic space and associated with the word label of the nearest neighbor in that space (*degus* in Figure 1b).

The fast mapping setting can be seen as a special case of the zero-shot task. Whereas for the latter our system assumes that all concepts have rich linguistic representations (i.e., representations estimated from a large corpus), in the case of the former, new concepts are assumed to be encounted in a limited linguistic context and therefore lacking rich linguistic representations. This is operationalized by constructing the text-based vector for these

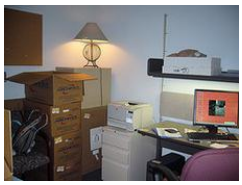



Figure 2: Images of *chair* as extracted from CIFAR-100 (left) and ESP (right).

concepts from a context of just a few occurrences. In this way, we simulate the first encounter of a learner with a concept that is new in both visual and linguistic terms.

4 Experimental Setup

4.1 Visual Datasets

CIFAR-100 The CIFAR-100 dataset (Krizhevsky, 2009) consists of 60,000 32x32 colour images (note the extremely small size) representing 100 distinct concepts, with 600 images per concept. The dataset covers a wide range of concrete domains and is organized into 20 broader categories. Table 1 lists the concepts used in our experiments organized by category.

ESP Our second dataset consists of 100K images from the ESP-Game data set, labeled through a "game with a purpose" (Von Ahn, 2006).[1] The ESP image tags form a vocabulary of 20,515 unique words. Unlike other datasets used for zero-shot learning, it covers adjectives and verbs in addition to nouns. On average, an image has 14 tags and a word appears as a tag for 70 images. Unlike the CIFAR-100 images, which were chosen specifically for image object recognition tasks (i.e., each image is clearly depicting a single object in the foreground), ESP contains a random selection of images from the Web. Consequently, objects do not appear in most images in their prototypical display, but rather as elements of complex scenes (see Figure 2). Thus, ESP constitutes a more realistic, and at the same time more challenging, simulation of how things are encountered in real life, testing the potentials of cross-modal mapping in dealing with the complex scenes that one would encounter in event recognition and caption generation tasks.

[1] http://www.cs.cmu.edu/~biglou/resources/

4.2 Visual Semantic Spaces

Image-based vectors are extracted using the unsupervised bag-of-visual-words (BoVW) representational architecture (Sivic and Zisserman, 2003; Csurka et al., 2004), that has been widely and successfully applied to computer vision tasks such as object recognition and image retrieval (Yang et al., 2007). First, low-level visual features (Szeliski, 2010) are extracted from a large collection of images and clustered into a set of "visual words". The low-level features of a specific image are then mapped to the corresponding visual words, and the image is represented by a count vector recording the number of occurrences of each visual word in it. We do not attempt any parameter tuning of the pipeline.

As low-level features, we use Scale Invariant Feature Transform (SIFT) features (Lowe, 2004). SIFT features are tailored to capture object parts and to be invariant to several image transformations such as rotation, illumination and scale change. These features are clustered into vocabularies of 5,000 (ESP) and 4,096 (CIFAR-100) visual words.[2] To preserve spatial information in the BoVW representation, we use the spatial pyramid technique (Lazebnik et al., 2006), which consists in dividing the image into several regions, computing BoVW vectors for each region and concatenating them. In particular, we divide ESP images into 16 regions and the smaller CIFAR-100 images into 4. The vectors resulting from region concatenation have dimensionality $5000 \times 16 = 80,000$ (ESP) and $4,096 \times 4 = 16,384$ (CIFAR-100), respectively. We apply Local Mutual Information (LMI, (Evert, 2005)) as weighting scheme and reduce the full co-occurrence space to 300 dimensions using the Singular Value Decomposition.

For CIFAR-100, we extract distinct visual vectors for single images. For ESP, given the size and amount of noise in this dataset, we build vectors for visual *concepts*, by normalizing and summing the BoVW vectors of all the images that have the relevant concept as a tag. Note that relevant literature (Pereira et al., 2010) has emphasized the importance of learners self-generating multiple views when faced with new objects. Thus, our multiple-image assumption should not be considered as problematic in the current setup.

[2] For selecting the size of the vocabulary size, we relied on standard settings found in the relevant literature (Bruni et al., 2014; Chatfield et al., 2011).

Category	Seen Concepts	Unseen (Test) Concepts
aquatic mammals	beaver, otter, seal, whale	dolphin
fish	ray, trout	shark
flowers	orchid, poppy, sunflower, tulip	rose
food containers	bottle, bowl, can ,plate	cup
fruit vegetable	apple, mushroom, pear	orange
household electrical devices	keyboard, lamp, telephone, television	clock
household furniture	chair, couch, table, wardrobe	bed
insects	bee, beetle, caterpillar, cockroach	butterfly
large carnivores	bear, leopard, lion, wolf	tiger
large man-made outdoor things	bridge, castle, house, road	skyscraper
large natural outdoor scenes	cloud, mountain, plain, sea	forest
large omnivores and herbivores	camel, cattle, chimpanzee, kangaroo	elephant
medium-sized mammals	fox, porcupine, possum, skunk	raccoon
non-insect invertebrates	crab, snail, spider, worm	lobster
people	baby, girl, man, woman	boy
reptiles	crocodile, dinosaur, snake, turtle	lizard
small mammals	hamster, mouse, rabbit, shrew	squirrel
vehicles 1	bicycle, motorcycle, train	bus
vehicles 2	rocket, tank, tractor	streetcar

Table 1: Concepts in our version of the CIFAR-100 data set

We implement the entire visual pipeline with VSEM, an open library for visual semantics (Bruni et al., 2013).[3]

4.3 Linguistic Semantic Spaces

For constructing the text-based vectors, we follow a standard pipeline in distributional semantics (Turney and Pantel, 2010) without tuning its parameters and collect co-occurrence statistics from the concatenation of ukWaC[4] and the Wikipedia, amounting to 2.7 billion tokens in total. Semantic vectors are constructed for a set of 30K target words (lemmas), namely the top 20K most frequent nouns, 5K most frequent adjectives and 5K most frequent verbs, and the same 30K lemmas are also employed as contextual elements. We collect co-occurrences in a symmetric context window of 20 elements around a target word. Finally, similarly to the visual semantic space, raw counts are transformed by applying LMI and then reduced to 300 dimensions with SVD.[5]

4.4 Cross-modal Mapping

The process of learning to map objects to the their word label is implemented by training a projection function $\mathrm{f}_{\mathrm{proj_{v\rightarrow w}}}$ from the visual onto the linguistic semantic space. For the learning, we use a set of N_s *seen* concepts for which we have both image-based visual representations $\mathbf{V}_s \in \mathbb{R}^{N_s \times d_v}$ and text-based linguistic representations $\mathbf{W}_s \in \mathbb{R}^{N_s \times d_w}$. The projection function is subject to an objective that aims at minimizing some cost function between the induced text-based representations $\hat{\mathbf{W}}_s \in \mathbb{R}^{N_s \times d_w}$ and the gold ones $\mathbf{W}_s$. The induced $\mathrm{f}_{\mathrm{proj_{v\rightarrow w}}}$ is then applied to the image-based representations $\mathbf{V}_u \in \mathbb{R}^{N_u \times d_v}$ of N_u unseen objects to transform them into text-based representations $\hat{\mathbf{W}}_u \in \mathbb{R}^{N_u \times d_w}$. We implement 4 alternative learning algorithms for inducing the cross-modal projection function $\mathrm{f}_{\mathrm{proj_{v\rightarrow w}}}$.

Linear Regression (lin) Our first model is a very simple linear mapping between the two modalities estimated by solving a least-squares problem. This method is similar to the one introduced by Mikolov et al. (2013a) for estimating a translation matrix, only solved analytically. In our setup, we can see the two different modalities as if they were different languages. By using least-squares regression, the projection function $\mathrm{f}_{\mathrm{proj_{v\rightarrow w}}}$ can be derived as

$$\mathrm{f}_{\mathrm{proj_{v\rightarrow w}}} = (\mathbf{V}_s^T \mathbf{V}_s)^{-1} \mathbf{V}_s^T \mathbf{W}_s \qquad (1)$$

Canonical Correlation Analysis (CCA) CCA (Hardoon et al., 2004; Hotelling, 1936) and variations thereof have been successfully used in the past for annotation of regions (Socher and Fei-Fei, 2010) and complete images (Hardoon et al., 2006; Hodosh et al., 2013). Given two paired observation matrices, in our case $\mathbf{V}_s$ and $\mathbf{W}_s$, CCA aims at capturing the linear relationship that exists between these variables. This is achieved by finding a pair of matrices, in our

[3] http://clic.cimec.unitn.it/vsem/

[4] http://wacky.sslmit.unibo.it

[5] We also experimented with the image- and text-based vectors of Socher et al. (2013), but achieved better performance with the reported setup.

case $\mathbf{C}_V \in \mathbb{R}^{d_v \times d}$ and $\mathbf{C}_W \in \mathbb{R}^{d_w \times d}$, such that the correlation between the projections of the two multidimensional variables into a common, lower-rank space is maximized. The resulting multimodal space has been shown to provide a good approximation to human concept similarity judgments (Silberer and Lapata, 2012). In our setup, after applying CCA on the two spaces $\mathbf{V}_s$ and $\mathbf{W}_s$, we obtain the two projection mappings onto the common space and thus our projection function can be derived as:

$$\mathrm{f}_{\mathrm{proj}_{\mathrm{v}\rightarrow\mathrm{w}}} = \mathbf{C}_V {\mathbf{C}_W}^{-1} \quad (2)$$

Singular Value Decomposition (SVD) SVD is the most widely used dimensionality reduction technique in distributional semantics (Turney and Pantel, 2010), and it has recently been exploited to combine visual and linguistic dimensions in the multimodal distributional semantic model of Bruni et al. (2014). SVD smoothing is also a way to infer values of unseen dimensions in partially incomplete matrices, a technique that has been applied to the task of inferring word tags of unannotated images (Hare et al., 2008). Assuming that the concept-representing rows of $\mathbf{V}_s$ and $\mathbf{W}_s$ are ordered in the same way, we apply the (k-truncated) SVD to the concatenated matrix $[\mathbf{V}_s\mathbf{W}_s]$, such that $[\hat{\mathbf{V}}_s\hat{\mathbf{W}}_\mathbf{s}] = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{Z}_k^T$ is a k-rank approximation of the original matrix.[6] The projection function is then:

$$\mathrm{f}_{\mathrm{proj}_{\mathrm{v}\rightarrow\mathrm{w}}} = \mathbf{Z}_k\mathbf{Z}_k^T \quad (3)$$

where the input is appropriately padded with 0s ($[\mathbf{V}_u\mathbf{0}_{Nu\times W}]$) and we discard the visual block of the output matrix $[\hat{\mathbf{V}}_u\hat{\mathbf{W}}_u]$.

Neural Network (NNet) The last model that we introduce is a neural network with one hidden layer. The projection function in this model can be described as:

$$\mathrm{f}_{\mathrm{proj}_{\mathrm{v}\rightarrow\mathrm{w}}} = \mathbf{\Theta}_{\mathrm{v}\rightarrow\mathrm{w}} \quad (4)$$

where $\mathbf{\Theta}_{v\rightarrow w}$ consists of the model weights $\boldsymbol{\theta}^{(1)} \in \mathbb{R}^{d_v \times h}$ and $\boldsymbol{\theta}^{(2)} \in \mathbb{R}^{h \times d_w}$ that map the input image-based vectors $\mathbf{V}_s$ first to the hidden layer and then to the output layer in order to obtain text-based vectors, i.e., $\hat{\mathbf{W}}_s = \sigma^{(2)}(\sigma^{(1)}(\mathbf{V}_s\boldsymbol{\theta}^{(1)})\boldsymbol{\theta}^{(2)})$, where $\sigma^{(1)}$ and $\sigma^{(2)}$ are the non-linear activation functions. We experimented with sigmoid, hyperbolic tangent and linear; hyperbolic tangent yielded the highest performance. The weights are estimated by minimizing the objective function

$$J(\mathbf{\Theta}_{\mathrm{v}\rightarrow\mathrm{w}}) = \frac{1}{2}(1 - sim(\mathbf{W}_s, \hat{\mathbf{W}}_s)) \quad (5)$$

where sim is some similarity function. In our experiments we used *cosine* as similarity function, so that $sim(\mathbf{A}, \mathbf{B}) = \frac{AB}{\|A\|\|B\|}$, thus penalizing parameter settings leading to a low cosine between the target linguistic representations $\mathbf{W}_s$ and those produced by the projection function $\hat{\mathbf{W}}_s$. The cosine has been widely used in the distributional semantic literature, and it has been shown to outperform Euclidean distance (Bullinaria and Levy, 2007).[7] Parameters were estimated with standard backpropagation and L-BFGS.

5 Results

Our experiments focus on the tasks of zero-shot learning (Sections 5.1 and 5.2) and fast mapping (Section 5.3). In both tasks, the projected vector of the unseen concept is labeled with the word associated to its cosine-based nearest neighbor vector in the corresponding semantic space.

For the zero-shot task we report the *accuracy* of retrieving the correct label among the top k neighbors from a semantic space populated with the *union of seen and unseen* concepts. For fast mapping, we report the *mean rank* of the correct concept among fast mapping candidates.

5.1 Zero-shot Learning in CIFAR-100

For this experiment, we use the intersection of our linguistic space with the concepts present in CIFAR-100, containing a total of 90 concepts. For each concept category, we treat all concepts but one as seen concepts (Table 1). The 71 seen concepts correspond to 42,600 distinct visual vectors and are used to induce the projection function. Table 2 reports results obtained by averaging the performance on the 11,400 distinct vectors of the 19 unseen concepts.

Our 4 models introduced in Section 4.4 are compared to a theoretically derived baseline **Chance** simulating selecting a label at random. For the neural network **NN**, we use prior knowledge

[6] We denote the right singular vectors matrix by $\mathbf{Z}$ instead of the customary $\mathbf{V}$ to avoid confusion with the visual matrix.

[7] We also experimented with the same objective function as Socher et al. (2013), however, our objective function yielded consistently better results in all experimental settings.

Model \ k	1	2	3	5	10	20
Chance	1.1	2.2	3.3	5.5	11.0	22.0
SVD	1.9	5.0	8.1	14.5	29.0	48.6
CCA	3.0	6.9	10.7	17.9	31.7	51.7
lin	2.4	6.4	10.5	18.7	33.0	55.0
NN	3.9	6.6	10.6	21.9	37.9	58.2

Table 2: Percentage accuracy among top k nearest neighbors on CIFAR-100.

about the number of concept categories to set the number of hidden units to 20 in order to avoid tuning of this parameter. For the **SVD** model, we set the number of dimensions to 300, a common choice in distributional semantics, coherent with the settings we used for the visual and linguistic spaces.

First and foremost, all 4 models outperform **Chance** by a large margin. Surprisingly, the very simple **lin** method outperforms both **CCA** and **SVD**. However, **NN**, an architecture that can capture more complex, non-linear relations in features across modalities, emerges as the best performing model, confirming on a larger scale the recent findings of Socher et al. (2013).

5.1.1 Concept Categorization

In order to gain qualitative insights into the performance of the projection process of **NN**, we attempt to investigate the role and interpretability of the *hidden layer*. We achieve this by looking at which visual concepts result in the *highest* hidden unit activation.[8] This is inspired by analogous qualitative analysis conducted in Topic Models (Griffiths et al., 2007), where "topics" are interpreted in terms of the words with the highest probability under each of them.

Table 3 presents both seen and unseen concepts corresponding to visual vectors that trigger the highest activation for a subset of hidden units. The table further reports, for each hidden unit, the "correct" unseen concept for the category of the top seen concepts, together with its rank in terms of activation of the unit. The analysis demonstrates that, although prior knowledge about categories was not explicitly used to train the network, the latter induced an organization of concepts into superordinate categories in which the hidden layer acts as a cross-modal concept categorization/organization system. When the induced projection function maps an object onto the linguistic space, the derived text vector will inherit a mixture of textual features from the concepts that activated the same hidden unit as the object. This suggests a bias towards seen concepts. Furthermore, in many cases of miscategorization, the concepts are still semantically coherent with the induced category, confirming that the projection function is indeed capturing a latent, cross-modal semantic space. A *squirrel*, although not a "*large* omnivore", is still an animal, while *butterflies* are not *flowers* but often feed on their nectar.

[8]For this post-hoc analysis, we include a sparsity parameter in the objective function of Equation 5 in order to get more interpretable results; hidden units are therefore maximally activated by a only few concepts.

Unseen Concept	Nearest Neighbors
tiger	cat, microchip, kitten, vet, pet
bike	spoke, wheel, brake, tyre, motorcycle
blossom	bud, leaf, jasmine, petal, dandelion
bakery	quiche, bread, pie, bagel, curry

Table 4: Top 5 neighbors in linguistic space after visual vector projection of 4 unseen concepts.

5.2 Zero-shot Learning in ESP

For this experiment, we focus on **NN**, the best performing model in the previous experiment. We use a set of approximately 9,500 concepts, the intersection of the ESP-based visual semantic space with the linguistic space. For tuning the number of hidden units of **NN**, we use the MEN-concrete dataset of Bruni et al. (2014). Finally, we randomly pick 70% of the concepts to induce the projection function $f_{proj_{v \rightarrow w}}$ and report results on the remaining 30%. Note that the search space for the correct label in this experiment is approximately 95 times larger than the one used for the experiment presented in Section 5.1.

Although our experimental setup differs from the one of Frome et al. (2013), thus preventing a direct comparison, the results reported in Table 5 are on a comparable scale to theirs. We note that previous work on zero-shot learning has used standard object recognition benchmarks. To the best of our knowledge, this is the first time this task has been performed on a dataset as noisy as ESP. Overall, the results suggest that cross-modal mapping could be applied in tasks where images exhibit a more complex structure, e.g., caption generation and event recognition.

	Seen Concepts	Unseen Concept	Rank of Correct Unseen Concept	CIFAR-100 Category
Unit 1	**sunflower**, **tulip**, pear	butterfly	2 (**rose**)	flowers
Unit 2	**cattle**, **camel**, **bear**	squirrel	2 (**elephant**)	large omnivores and herbivores
Unit 3	**castle**, **bridge**, **house**	bus	4 (**skyscraper**)	large man-made outdoor things
Unit 4	**man**, **girl**, **baby**	**boy**	1	people
Unit 5	**motorcycle**, **bicycle**, tractor	streetcar	2 (**bus**)	vehicles 1
Unit 6	**sea**, **plain**, **cloud**	**forest**	1	large natural outdoor scenes
Unit 7	**chair**, **couch**, **table**	**bed**	1	household furniture
Unit 8	**plate**, **bowl**, **can**	clock	3 (**cup**)	food containers
Unit 9	**apple**, **pear**, **mushroom**	**orange**	1	fruit and vegetables

Table 3: Categorization induced by the hidden layer of the **NN**; concepts belonging in the same CIFAR-100 categories, reported in the last column, are marked in bold. Example: Unit 1 receives the highest activation during training by the category *flowers* and at test time by *butterfly*, belonging to *insects*. The same unit receives the second highest activation by the "correct" test concept, the *flower rose*.

Model \ k	1	2	5	10	50
Chance	0.01	0.02	0.05	0.10	0.5
NN	0.8	1.9	5.6	9.7	30.9

Table 5: Percentage accuracy among top k nearest neighbors on ESP.

5.3 Fast Mapping in ESP

In this section, we aim at simulating a fast mapping scenario in which the learner has been just exposed to a new concept, and thus has limited linguistic evidence for that concept. We operationalize this by considering the 34 concrete concepts introduced by Frassinelli and Keller (2012), and deriving their text-based representations from just a few sentences randomly picked from the corpus. Concretely, we implement 5 models: **context 1**, **context 5**, **context 10**, **context 20** and **context full**, where the name of the model denotes the number of sentences used to construct the text-based representations. The derived vectors were reduced with the same SVD projection induced from the complete corpus. Cross-modal mapping is done via **NN**.

The zero-shot framework leads us to frame fast mapping as the task of projecting visual representations of new objects onto language space for retrieving their word labels (v $\rightarrow$ w). This mapping from visual to textual representations is arguably a more plausible task than *vice versa*. If we think about how linguistic reference is acquired, a scenario in which a learner *first* encounters a new object and *then* seeks its reference in the language of the surrounding environment (e.g., adults having a conversation, the text of a book with an illustration of an unknown object) is very natural. Furthermore, since not all new concepts in the linguistic environment refer to new objects (they might denote abstract concepts or out-of-scene objects), it seems more reasonable for the learner to be more alerted to linguistic cues about a recently-spotted new object than *vice versa*. Moreover, once the learner observes a new object, she can easily construct a full visual representation for it (and the acquisition literature has shown that humans are wired for good object segmentation and recognition (Spelke, 1994)) – the more challenging task is to scan the ongoing and very ambiguous linguistic communication for contexts that might be relevant and informative about the new object. However, fast mapping is often described in the psychological literature as the opposite task: The learner is exposed to a new word in context and has to search for the right object referring to it. We implement this second setup (w $\rightarrow$ v) by training the projection function $\mathrm{f}_{\mathrm{proj}_{\mathrm{w}\rightarrow\mathrm{v}}}$ which maps linguistic vectors to visual ones. The adaptation of **NN** is straightforward; the new objective function is derived as

$$J(\mathbf{\Theta}_{\mathrm{w}\rightarrow\mathrm{v}}) = \frac{1}{2}(1 - sim(\mathbf{V}_s, \hat{\mathbf{V}}_s)) \qquad (6)$$

where $\hat{\mathbf{V}}_s = \sigma^{(2)}(\sigma^{(1)}(\mathbf{W}_s\boldsymbol{\theta}^{(1)})\boldsymbol{\theta}^{(2)})$, $\boldsymbol{\theta}^{(1)} \in \mathbb{R}^{d_w \times h}$ and $\boldsymbol{\theta}^{(2)} \in \mathbb{R}^{h \times d_v}$.

Table 7 presents the results. Not surprisingly, performance increases with the number of sentences that are used to construct the textual representations. Furthermore, all models perform better than **Chance**, including those that are based on just **1** or **5** sentences. This suggests that the system can make reasonable inferences about object-word connections even when linguistic evidence is very scarce.

Regarding the sources of error, a qualitative analysis of predicted word labels and objects as

v→w	w→v
cooker→potato	dishwasher→ corkscrew
clarinet→ drum	potato→ corn
gorilla→ elephant	guitar→ violin
scooter→ car	scarf→ trouser

Table 6: Top-ranked concepts in cases where the gold concepts received numerically high ranks.

Context \ Mapping	v → w	w → v
Chance	17	17
context 1	12.6	14.5
context 5	8.08	13.29
context 10	7.29	13.44
context 20	6.02	12.17
context full	5.52	5.88

Table 7: Mean rank results averaged across 34 concepts when mapping an image-based vector and retrieving its linguistic neighbors (v → w) as well as when mapping a text-based vector and retrieving its visual neighbors (w → v). Lower numbers cue better performance.

presented in Table 6 suggests that both textual and visual representations, although capturing relevant "topical" or "domain" information, are not enough to single out the properties of the target concept. As an example, the textual vector of *dishwasher* contains kitchen-related dimensions such as ⟨**fridge**, **oven**, **gas**, **hob**, ..., **sink**⟩. After projecting onto the visual space, its nearest visual neighbours are the visual ones of the same-domain concepts *corkscrew* and *kettle*. The latter is shown in Figure 3a, with a *gas hob* well in evidence. As a further example, the visual vector for *cooker* is extracted from pictures such as the one in Figure 3b. Not surprisingly, when projecting it onto the linguistic space, the nearest neighbours are other kitchen-related terms, i.e., *potato* and *dishwasher*.

6 Conclusion

At the outset of this work, we considered the problem of linking purely language-based distributional semantic spaces with objects in the visual world by means of cross-modal mapping. We compared recent models for this task both on a benchmark object recognition dataset and on a more realistic and noisier dataset covering a wide range of concepts. The neural network architecture emerged as the best performing approach, and our qualitative analysis revealed that it induced a categorical organization of concepts. Most importantly, our results suggest the viability of cross-modal mapping for grounded word-meaning acquisition in a simulation of fast mapping.

Given the success of **NN**, we plan to experiment in the future with more sophisticated neural network architectures inspired by recent work in machine translation (Gao et al., 2013) and multimodal deep learning (Srivastava and Salakhutdinov, 2012). Furthermore, we intend to adopt *visual attributes* (Farhadi et al., 2009; Silberer et al., 2013) as visual representations, since they should allow a better understanding of how cross-modal mapping works, thanks to their linguistic interpretability. The error analysis in Section 5.3 suggests that automated *localization* techniques (van de Sande et al., 2011), distinguishing an object from its surroundings, might drastically improve mapping accuracy. Similarly, in the textual domain, models that extract collocates of a word that are more likely to denote conceptual properties (Kelly et al., 2012) might lead to more informative and discriminative linguistic vectors. Finally, the lack of large child-directed speech corpora constrained the experimental design of fast mapping simulations; we plan to run more realistic experiments with true nonce words and using source corpora (e.g., the Simple Wikipedia, child stories, portions of CHILDES) that contain sentences more akin to those a child might effectively hear or read in her word-learning years.

(a) A kettle

(b) A cooker

Figure 3: Two images from ESP.

Acknowledgments

We thank Adam Liška for helpful discussions and the 3 anonymous reviewers for useful comments. This work was supported by ERC 2011 Starting Independent Research Grant n. 283554 (COMPOSES).

References

Barbara Abbott. 2010. *Reference*. Oxford University Press, Oxford, UK.

Paul Bloom. 2000. *How Children Learn the Meanings of Words*. MIT Press, Cambridge, MA.

S. R. K. Branavan, Harr Chen, Luke S. Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of ACL/IJCNLP*, pages 82–90.

Elia Bruni, Ulisse Bordignon, Adam Liska, Jasper Uijlings, and Irina Sergienya. 2013. Vsem: An open library for visual semantics representation. In *Proceedings of ACL*, Sofia, Bulgaria.

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

John Bullinaria and Joseph Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, 39:510–526.

Susan Carey and Elsa Bartlett. 1978. Acquiring a single new word. *Papers and Reports on Child Language Development*, 15:17–29.

Susan Carey. 1978. The child as a word learner. In M. Halle, J. Bresnan, and G. Miller, editors, *Linguistics Theory and Psychological Reality*. MIT Press, Cambridge, MA.

Ken Chatfield, Victor Lempitsky, Andrea Vedaldi, and Andrew Zisserman. 2011. The devil is in the details: an evaluation of recent feature encoding methods. In *Proceedings of BMVC*, Dundee, UK.

David Chen and Raymond Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of AAAI*, pages 859–865, San Francisco, CA.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*, pages 160–167, Helsinki, Finland.

Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. 2004. Visual categorization with bags of keypoints. In *In Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, Prague, Czech Republic.

Stefan Evert. 2005. *The Statistics of Word Cooccurrences*. Ph.D dissertation, Stuttgart University.

Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. 2009. Describing objects by their attributes. In *Proceedings of CVPR*, pages 1778–1785, Miami Beach, FL.

Afsaneh Fazly, Afra Alishahi, and Suzanne Stevenson. 2010. A probabilistic computational model of cross-situational word learning. *Cognitive Science*, 34:1017–1063.

Yansong Feng and Mirella Lapata. 2010. Visual information in semantic representation. In *Proceedings of HLT-NAACL*, pages 91–99, Los Angeles, CA.

Diego Frassinelli and Frank Keller. 2012. The plausibility of semantic properties generated by a distributional model: Evidence from a visual world experiment. In *Proceedings of CogSci*, pages 1560–1565.

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

Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. 2013. Learning semantic representations for the phrase translation model. *arXiv preprint arXiv:1312.0482*.

Tom Griffiths, Mark Steyvers, and Josh Tenenbaum. 2007. Topics in semantic representation. *Psychological Review*, 114:211–244.

David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. 2004. Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, 16(12):2639–2664.

David R Hardoon, Craig Saunders, Sandor Szedmak, and John Shawe-Taylor. 2006. A correlation approach for automatic image annotation. In *Advanced Data Mining and Applications*, pages 681–692. Springer.

Jonathon Hare, Sina Samangooei, Paul Lewis, and Mark Nixon. 2008. Semantic spaces revisited: Investigating the performance of auto-annotation and semantic retrieval using semantic spaces. In *Proceedings of CIVR*, pages 359–368, Niagara Falls, Canada.

Stevan Harnad. 1990. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1-3):335–346.

Tracy Heibeck and Ellen Markman. 1987. Word learning in children: an examination of fast mapping. *Child Development*, 58:1021–1024.

Micah Hodosh, Peter Young, and Julia Hockenmaier. 2013. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research*, 47:853–899.

Harold Hotelling. 1936. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377.

Colin Kelly, Barry Devereux, and Anna Korhonen. 2012. Semi-supervised learning for automatic conceptual property extraction. In *Proceedings of the 3rd Workshop on Cognitive Modeling and Computational Linguistics*, pages 11–20, Montreal, Canada.

Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proceedings of NIPS*, pages 1106–1114.

Alex Krizhevsky. 2009. Learning multiple layers of features from tiny images. Master's thesis.

Thomas Landauer and Susan Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240.

Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. 2006. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of CVPR*, pages 2169–2178, Washington, DC.

Alessandro Lenci. 2008. Distributional approaches in linguistic and cognitive research. *Italian Journal of Linguistics*, 20(1):1–31.

Chee Wee Leong and Rada Mihalcea. 2011. Going beyond text: A hybrid image-text approach for measuring word relatedness. In *Proceedings of IJCNLP*, pages 1403–1407.

David Lowe. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2).

Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods*, 28:203–208.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119, Lake Tahoe, Nevada.

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

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

Alfredo F Pereira, Karin H James, Susan S Jones, and Linda B Smith. 2010. Early biases and developmental changes in self-generated object views. *Journal of vision*, 10(11).

Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. 2013. Grounding action descriptions in videos. *Transactions of the Association for Computational Linguistics*, 1:25–36.

John Searle. 1984. *Minds, Brains and Science*. Harvard University Press, Cambridge, MA.

Carina Silberer and Mirella Lapata. 2012. Grounded models of semantic representation. In *Proceedings of EMNLP*, pages 1423–1433, Jeju, Korea.

Carina Silberer, Vittorio Ferrari, and Mirella Lapata. 2013. Models of semantic representation with visual attributes. In *Proceedings of ACL*, pages 572–582, Sofia, Bulgaria.

Jeffrey Siskind. 1996. A computational study of cross-situational techniques for learning word-to-meaning mappings. *Cognition*, 61:39–91.

Josef Sivic and Andrew Zisserman. 2003. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of ICCV*, pages 1470–1477, Nice, France.

Richard Socher and Li Fei-Fei. 2010. Connecting modalities: Semi-supervised segmentation and annotation of images using unaligned text corpora. In *Proceedings of CVPR*, pages 966–973.

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

Elizabeth Spelke. 1994. Initial knowledge: Six suggestions. *Cognition*, 50:431–445.

Nitish Srivastava and Ruslan Salakhutdinov. 2012. Multimodal learning with deep boltzmann machines. In *Proceedings of NIPS*, pages 2231–2239.

Richard Szeliski. 2010. *Computer Vision : Algorithms and Applications*. Springer, Berlin.

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

Koen van de Sande, Jasper Uijlings, Theo Gevers, and Arnold Smeulders. 2011. Segmentation as selective search for object recognition. In *Proceedings of ICCV*, pages 1879–1886, Barcelona, Spain.

Luis Von Ahn. 2006. Games with a purpose. *Computer*, 29(6):92–94.

Jun Yang, Yu-Gang Jiang, Alexander Hauptmann, and Chong-Wah Ngo. 2007. Evaluating bag-of-visual-words representations in scene classification. In James Ze Wang, Nozha Boujemaa, Alberto Del Bimbo, and Jia Li, editors, *Multimedia Information Retrieval*, pages 197–206. ACM.

Haonan Yu and Jeffrey Siskind. 2013. Grounded language learning from video described with sentences. In *Proceedings of ACL*, pages 53–63, Sofia, Bulgaria.

Semantic Parsing via Paraphrasing

Jonathan Berant
Stanford University
joberant@stanford.edu

Percy Liang
Stanford University
pliang@cs.stanford.edu

Abstract

A central challenge in semantic parsing is handling the myriad ways in which knowledge base predicates can be expressed. Traditionally, semantic parsers are trained primarily from text paired with knowledge base information. Our goal is to exploit the much larger amounts of raw text not tied to any knowledge base. In this paper, we turn semantic parsing on its head. Given an input utterance, we first use a simple method to deterministically generate a set of candidate logical forms with a canonical realization in natural language for each. Then, we use a paraphrase model to choose the realization that best paraphrases the input, and output the corresponding logical form. We present two simple paraphrase models, an *association* model and a *vector space* model, and train them jointly from question-answer pairs. Our system PARASEMPRE improves state-of-the-art accuracies on two recently released question-answering datasets.

1 Introduction

We consider the semantic parsing problem of mapping natural language utterances into logical forms to be executed on a knowledge base (KB) (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Wong and Mooney, 2007; Kwiatkowski et al., 2010). Scaling semantic parsers to large knowledge bases has attracted substantial attention recently (Cai and Yates, 2013; Berant et al., 2013; Kwiatkowski et al., 2013), since it drives applications such as question answering (QA) and information extraction (IE).

Semantic parsers need to somehow associate natural language phrases with logical predicates, e.g., they must learn that the constructions *"What does X do for a living?"*, *"What is X's profession?"*, and *"Who is X?"*, should all map to the logical predicate `Profession`. To learn these mappings, traditional semantic parsers use data which pairs natural language with the KB. However, this leaves untapped a vast amount of text not related to the KB. For instance, the utterances *"Where is ACL in 2014?"* and *"What is the location of ACL 2014?"* cannot be used in traditional semantic parsing methods, since the KB does not contain an entity `ACL2014`, but this pair clearly contains valuable linguistic information. As another reference point, out of 500,000 relations extracted by the ReVerb Open IE system (Fader et al., 2011), only about 10,000 can be aligned to Freebase (Berant et al., 2013).

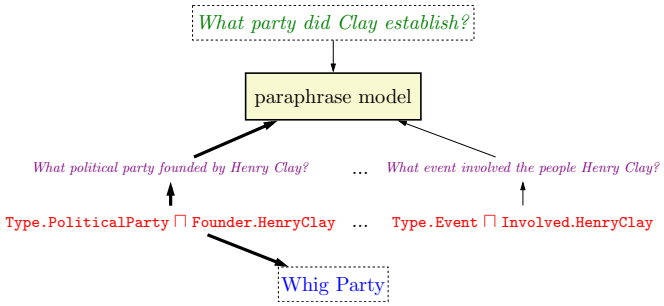


Figure 1: Semantic parsing via paraphrasing: For each candidate logical form (in red), we generate canonical utterances (in purple). The model is trained to paraphrase the input utterance (in green) into the canonical utterances associated with the correct denotation (in blue).

In this paper, we present a novel approach for semantic parsing based on paraphrasing that can exploit large amounts of text not covered by the KB (Figure 1). Our approach targets factoid questions with a modest amount of compositionality. Given an input utterance, we first use a simple deterministic procedure to construct a manageable set of candidate logical forms (ideally, we would generate canonical utterances for all possible logical forms, but this is intractable). Next, we heuris-

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1415–1425,
Baltimore, Maryland, USA, June 23-25 2014.

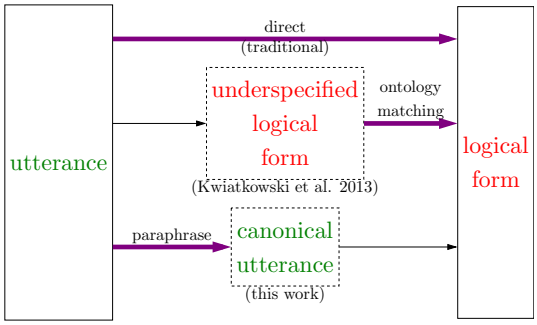


Figure 2: The main challenge in semantic parsing is coping with the *mismatch* between language and the KB. (a) Traditionally, semantic parsing maps utterances directly to logical forms. (b) Kwiatkowski et al. (2013) map the utterance to an underspecified logical form, and perform ontology matching to handle the mismatch. (c) We approach the problem in the other direction, generating canonical utterances for logical forms, and use paraphrase models to handle the mismatch.

tically generate *canonical utterances* for each logical form based on the text descriptions of predicates from the KB. Finally, we choose the canonical utterance that best paraphrases the input utterance, and thereby the logical form that generated it. We use two complementary paraphrase models: an *association model* based on aligned phrase pairs extracted from a monolingual parallel corpus, and a *vector space model*, which represents each utterance as a vector and learns a similarity score between them. The entire system is trained jointly from question-answer pairs only.

Our work relates to recent lines of research in semantic parsing and question answering. Kwiatkowski et al. (2013) first maps utterances to a domain-independent *intermediate logical form*, and then performs ontology matching to produce the final logical form. In some sense, we approach the problem from the opposite end, using an *intermediate utterance*, which allows us to employ paraphrasing methods (Figure 2). Fader et al. (2013) presented a QA system that maps questions onto simple queries against Open IE extractions, by learning paraphrases from a large monolingual parallel corpus, and performing a single paraphrasing step. We adopt the idea of using paraphrasing for QA, but suggest a more general paraphrase model and work against a formal KB (Freebase).

We apply our semantic parser on two datasets: WEBQUESTIONS (Berant et al., 2013), which contains 5,810 question-answer pairs with common questions asked by web users; and FREE917 (Cai and Yates, 2013), which has 917 questions manually authored by annotators. On WEBQUESTIONS, we obtain a relative improvement of 12% in accuracy over the state-of-the-art, and on FREE917 we match the current best performing system. The source code of our system PARASEMPRE is released at `http://www-nlp.stanford.edu/software/sempre/`.

2 Setup

Our task is as follows: Given (i) a knowledge base $\mathcal{K}$, and (ii) a training set of question-answer pairs $\{(x_i, y_i)\}_{i=1}^n$, output a semantic parser that maps new questions x to answers y via latent logical forms z. Let $\mathcal{E}$ denote a set of *entities* (e.g., `BillGates`), and let $\mathcal{P}$ denote a set of *properties* (e.g., `PlaceOfBirth`). A *knowledge base* $\mathcal{K}$ is a set of *assertions* $(e_1, p, e_2) \in \mathcal{E} \times \mathcal{P} \times \mathcal{E}$ (e.g., (`BillGates`, `PlaceOfBirth`, `Seattle`)). We use the Freebase KB (Google, 2013), which has 41M entities, 19K properties, and 596M assertions.

To query the KB, we use a logical language called *simple λ-DCS*. In simple λ-DCS, an entity (e.g., `Seattle`) is a unary predicate (i.e., a subset of $\mathcal{E}$) denoting a singleton set containing that entity. A property (which is a binary predicate) can be joined with a unary predicate; e.g., `Founded.Microsoft` denotes the entities that are Microsoft founders. In `PlaceOfBirth.Seattle` $\sqcap$ `Founded.Microsoft`, an intersection operator allows us to denote the set of Seattle-born Microsoft founders. A reverse operator reverses the order of arguments: **R**`[PlaceOfBirth].BillGates` denotes Bill Gates's birthplace (in contrast to `PlaceOfBirth.Seattle`). Lastly, `count(Founded.Microsoft)` denotes set cardinality, in this case, the number of Microsoft founders. The denotation of a logical form z with respect to a KB $\mathcal{K}$ is given by $[\![z]\!]_{\mathcal{K}}$. For a formal description of simple λ-DCS, see Liang (2013) and Berant et al. (2013).

3 Model overview

We now present the general framework for semantic parsing via paraphrasing, including the model and the learning algorithm. In Sections 4 and 5, we provide the details of our implementation.

Canonical utterance construction Given an utterance x and the KB, we construct a set of candi-

date logical forms $\mathcal{Z}_x$, and then for each $z \in \mathcal{Z}_x$ generate a small set of canonical natural language utterances $\mathcal{C}_z$. Our goal at this point is only to generate a manageable set of logical forms containing the correct one, and then generate an appropriate canonical utterance from it. This strategy is feasible in factoid QA where compositionality is low, and so the size of $\mathcal{Z}_x$ is limited (Section 4).

Paraphrasing We score the canonical utterances in $\mathcal{C}_z$ with respect to the input utterance x using a paraphrase model, which offers two advantages. First, the paraphrase model is decoupled from the KB, so we can train it from large text corpora. Second, natural language utterances often do not express predicates explicitly, e.g., the question *"What is Italy's money?"* expresses the binary predicate `CurrencyOf` with a possessive construction. Paraphrasing methods are well-suited for handling such text-to-text gaps. Our framework accommodates any paraphrasing method, and in this paper we propose an *association model* that learns to associate natural language phrases that co-occur frequently in a monolingual parallel corpus, combined with a *vector space model*, which learns to score the similarity between vector representations of natural language utterances (Section 5).

Model We define a discriminative log-linear model that places a probability distribution over pairs of logical forms and canonical utterances (c, z), given an utterance x:

$$p_\theta(c, z \mid x) = \frac{\exp\{\phi(x, c, z)^\top \theta\}}{\sum_{z' \in \mathcal{Z}_x, c' \in \mathcal{C}_z} \exp\{\phi(x, c', z')^\top \theta\}},$$

where $\theta \in \mathbb{R}^b$ is the vector of parameters to be learned, and $\phi(x, c, z)$ is a feature vector extracted from the input utterance x, the canonical utterance c, and the logical form z. Note that the candidate set of logical forms $\mathcal{Z}_x$ and canonical utterances $\mathcal{C}_x$ are constructed during the canonical utterance construction phase.

The model score decomposes into two terms:

$$\phi(x, c, z)^\top \theta = \phi_{\text{pr}}(x, c)^\top \theta_{\text{pr}} + \phi_{\text{lf}}(x, z)^\top \theta_{\text{lf}},$$

where the parameters θ_{pr} define the paraphrase model (Section 5), which is based on features extracted from text only (the input and canonical utterance). The parameters θ_{lf} correspond to semantic parsing features based on the logical form and input utterance, and are briefly described in this section.

Many existing paraphrase models introduce latent variables to describe the *derivation* of c from x, e.g., with transformations (Heilman and Smith, 2010; Stern and Dagan, 2011) or alignments (Haghighi et al., 2005; Das and Smith, 2009; Chang et al., 2010). However, we opt for a simpler paraphrase model without latent variables in the interest of efficiency.

Logical form features The parameters θ_{lf} correspond to the following features adopted from Berant et al. (2013). For a logical form z, we extract the size of its denotation $[\![z]\!]_{\mathcal{K}}$. We also add all binary predicates in z as features. Moreover, we extract a popularity feature for predicates based on the number of instances they have in $\mathcal{K}$. For Freebase entities, we extract a popularity feature based on the entity frequency in an entity linked subset of Reverb (Lin et al., 2012). Lastly, Freebase formulas have types (see Section 4), and we conjoin the type of z with the first word of x, to capture the correlation between a word (e.g., *"where"*) with the Freebase type (e.g., `Location`).

Learning As our training data consists of question-answer pairs (x_i, y_i), we maximize the log-likelihood of the correct answer. The probability of an answer y is obtained by marginalizing over canonical utterances c and logical forms z whose denotation is y. Formally, our objective function $\mathcal{O}(\theta)$ is as follows:

$$\mathcal{O}(\theta) = \sum_{i=1}^{n} \log p_\theta(y_i \mid x_i) - \lambda \|\theta\|_1,$$

$$p_\theta(y \mid x) = \sum_{z \in \mathcal{Z}_x : y = [\![z]\!]_{\mathcal{K}}} \sum_{c \in \mathcal{C}_z} p_\theta(c, z \mid x).$$

The strength λ of the L_1 regularizer is set based on cross-validation. We optimize the objective by initializing the parameters θ to zero and running AdaGrad (Duchi et al., 2010). We approximate the set of pairs of logical forms and canonical utterances with a beam of size 2,000.

4 Canonical utterance construction

We construct canonical utterances in two steps. Given an input utterance x, we first construct a set of logical forms $\mathcal{Z}_x$, and then generate canonical utterances from each $z \in \mathcal{Z}_x$. Both steps are performed with a small and simple set of deterministic rules, which suffices for our datasets, as

they consist of factoid questions with a modest amount of compositional structure. We describe these rules below for completeness. Due to its soporific effect though, we advise the reader to skim it quickly.

Candidate logical forms We consider logical forms defined by a set of templates, summarized in Table 1. The basic template is a join of a binary and an entity, where a binary can either be one property $p.e$ (#1 in the table) or two properties $p_1.p_2.e$ (#2). To handle cases of events involving multiple arguments (e.g., *"Who did Brad Pitt play in Troy?"*), we introduce the template $p.(p_1.e_1 \sqcap p_2.e_2)$ (#3), where the main event is modified by more than one entity. Logical forms can be further modified by a unary "filter", e.g., the answer to *"What composers spoke French?"* is a set of composers, i.e., a subset of all people (#4). Lastly, we handle aggregation formulas for utterances such as *"How many teams are in the NCAA?"* (#5).

To construct candidate logical forms $\mathcal{Z}_x$ for a given utterance x, our strategy is to find an entity in x and grow the logical form from that entity. As we show later, this procedure actually produces a set with better coverage than constructing logical forms recursively from spans of x, as is done in traditional semantic parsing. Specifically, for every span of x, we take at most 10 entities whose Freebase descriptions approximately match the span. Then, we join each entity e with all type-compatible[1] binaries b, and add these logical forms to $\mathcal{Z}_x$ (#1 and #2).

To construct logical forms with multiple entities (#3) we do the following: For any logical form $z = p.p_1.e_1$, where p_1 has type signature $(t_1, *)$, we look for other entities e_2 that were matched in x. Then, we add the logical form $p.(p_1.e_1 \sqcap p_2.e_2)$, if there exists a binary p_2 with a compatible type signature (t_1, t_2), where t_2 is one of e_2's types. For example, for the logical form `Character.Actor.BradPitt`, if we match the entity `Troy` in x, we obtain `Character.(Actor.BradPitt ⊓ Film.Troy)`. We further modify logical forms by intersecting with a unary filter (#4): given a formula z with some Freebase type (e.g., `People`), we look at all Freebase sub-types t (e.g., `Composer`), and check whether one of their Freebase descriptions (e.g., *"composer"*) appears in x. If so, we add the formula `Type`$.t \sqcap z$ to $\mathcal{Z}_x$. Finally, we check whether x is an aggregation formula by identifying whether it starts with phrases such as *"how many"* or *"number of"* (#5).

On WEBQUESTIONS, this results in 645 formulas per utterance on average. Clearly, we can increase the expressivity of this step by expanding the template set. For example, we could handle superlative utterances (*"What NBA player is tallest?"*) by adding a template with an `argmax` operator.

Utterance generation While mapping general language utterances to logical forms is hard, we observe that it is much easier to generate a *canonical* natural language utterances of our choice given a logical form. Table 2 summarizes the rules used to generate canonical utterances from the template $p.e$. Questions begin with a question word, are followed by the Freebase description of the expected answer type ($d(t)$), and followed by Freebase descriptions of the entity ($d(e)$) and binary ($d(p)$). To fill in auxiliary verbs, determiners, and prepositions, we parse the description $d(p)$ into one of NP, VP, PP, or NP VP. This determines the generation rule to be used.

Each Freebase property p has an explicit property p' equivalent to the reverse $\mathbf{R}[p]$ (e.g., `ContainedBy` and $\mathbf{R}$[`Contains`]). For each logical form z, we also generate using equivalent logical forms where p is replaced with $\mathbf{R}[p']$. Reversed formulas have different generation rules, since entities in these formulas are in the subject position rather than object position.

We generate the description $d(t)$ from the Freebase description of the type of z (this handles #4). For the template $p_1.p_2.e$ (#2), we have a similar set of rules, which depends on the syntax of $d(p_1)$ and $d(p_2)$ and is omitted for brevity. The template $p.(p_1.e_1 \sqcap p_2.e_2)$ (#3) is generated by appending the prepositional phrase `in` $d(e_2)$, e.g, *"What character is the character of Brad Pitt in Troy?"*. Lastly, we choose the question phrase *"How many"* for aggregation formulas (#5), and *"What"* for all other formulas.

We also generate canonical utterances using an alignment lexicon, released by Berant et al. (2013), which maps text phrases to Freebase binary predicates. For a binary predicate b mapped from text phrase $d(b)$, we generate the utterance

[1] Entities in Freebase are associated with a set of types, and properties have a type signature (t_1, t_2) We use these types to compute an expected type t for any logical form z.

#	Template	Example	Question
1	$p.e$	`Directed.TopGun`	*Who directed Top Gun?*
2	$p_1.p_2.e$	`Employment.EmployerOf.SteveBalmer`	*Where does Steve Balmer work?*
3	$p.(p_1.e_1 \sqcap p_2.e_2)$	`Character.(Actor.BradPitt` $\sqcap$ `Film.Troy)`	*Who did Brad Pitt play in Troy?*
4	`Type.`$t \sqcap z$	`Type.Composer` $\sqcap$ `SpeakerOf.French`	*What composers spoke French?*
5	`count(`z`)`	`count(BoatDesigner.NatHerreshoff)`	*How many ships were designed by Nat Herreshoff?*

Table 1: Logical form templates, where p, p_1, p_2 are Freebase properties, e, e_1, e_2 are Freebase entities, t is a Freebase type, and z is a logical form.

	$d(p)$ Categ.	Rule	Example
$p.e$	NP	`WH` $d(t)$ `has` $d(e)$ `as` NP ?	*What* ***election contest*** *has* ***George Bush*** *as* ***winner?***
	VP	`WH` $d(t)$ `(AUX)` VP $d(e)$?	*What* ***radio station serves area New-York****?*
	PP	`WH` $d(t)$ PP $d(e)$?	*What* ***beer from region Argentina****?*
	NP VP	`WH` $d(t)$ VP `the` NP $d(e)$?	*What* ***mass transportation system served*** *the* ***area Berlin****?*
$\mathbf{R}(p).e$	NP	`WH` $d(t)$ `is the` NP `of` $d(e)$?	*What* ***location*** *is the* ***place of birth*** *of* ***Elvis Presley****?*
	VP	`WH` $d(t)$ `AUX` $d(e)$ VP ?	*What* ***film*** *is* ***Brazil featured in****?*
	PP	`WH` $d(t)$ $d(e)$ PP ?	*What* ***destination Spanish steps near travel destination****?*
	NP VP	`WH` NP `is` VP `by` $d(e)$?	*What* ***structure*** *is* ***designed*** *by Herod?*

Table 2: Generation rules for templates of the form $p.e$ and $\mathbf{R}[p].e$ based on the syntactic category of the property description. Freebase descriptions for the type, entity, and property are denoted by $d(t)$, $d(e)$ and $d(p)$ respectively. The surface form of the auxiliary `AUX` is determined by the POS tag of the verb inside the VP tree.

`WH` $d(t)$ $d(b)$ $d(e)$ `?`. On the WEBQUESTIONS dataset, we generate an average of 1,423 canonical utterances c per input utterance x. In Section 6, we show that an even simpler method of generating canonical utterances by concatenating Freebase descriptions hurts accuracy by only a modest amount.

5 Paraphrasing

Once the candidate set of logical forms paired with canonical utterances is constructed, our problem is reduced to scoring pairs (c, z) based on a paraphrase model. The NLP paraphrase literature is vast and ranges from simple methods employing surface features (Wan et al., 2006), through vector space models (Socher et al., 2011), to latent variable models (Das and Smith, 2009; Wang and Manning, 2010; Stern and Dagan, 2011).

In this paper, we focus on two paraphrase models that emphasize simplicity and efficiency. This is important since for each question-answer pair, we consider thousands of canonical utterances as potential paraphrases. In contrast, traditional paraphrase detection (Dolan et al., 2004) and Recognizing Textual Entailment (RTE) tasks (Dagan et al., 2013) consider examples consisting of only a single pair of candidate paraphrases.

Our paraphrase model decomposes into an *association model* and a *vector space model*:

$$\phi_{\text{pr}}(x, c)^\top \theta_{\text{pr}} = \phi_{\text{as}}(x, c)^\top \theta_{\text{as}} + \phi_{\text{vs}}(x, c)^\top \theta_{\text{vs}}.$$

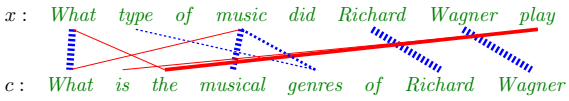


Figure 3: Token associations extracted for a paraphrase pair. Blue and dashed (red and solid) indicate positive (negative) score. Line width is proportional to the absolute value of the score.

5.1 Association model

The goal of the association model is to determine whether x and c contain phrases that are likely to be paraphrases. Given an utterance $x = \langle x_0, x_1, .., x_{n-1} \rangle$, we denote by $x_{i:j}$ the span from token i to token j. For each pair of utterances (x, c), we go through all spans of x and c and identify a set of pairs of potential paraphrases $(x_{i:j}, c_{i':j'})$, which we call *associations*. (We will describe how associations are identified shortly.) We then define features on each association; the weighted combination of these features yields a score. In this light, associations can be viewed as soft paraphrase rules. Figure 3 presents examples of associations extracted from a paraphrase pair and visualizes the learned scores. We can see that our model learns a positive score for associating *"type"* with *"genres"*, and a negative score for associating *"is"* with *"play"*.

We define associations in x and c primarily by looking up phrase pairs in a phrase table constructed using the PARALEX corpus (Fader et al., 2013). PARALEX is a large monolingual parallel

Category	Description
Assoc.	lemma($x_{i:j}$) $\wedge$ lemma($c_{i':j'}$) pos($x_{i:j}$) $\wedge$ pos($c_{i':j'}$) lemma($x_{i:j}$) = lemma($c_{i':j'}$)? pos($x_{i:j}$) = pos($c_{i':j'}$)? lemma($x_{i:j}$) and lemma($c_{i':j'}$) are synonyms? lemma($x_{i:j}$) and lemma($c_{i':j'}$) are derivations?
Deletions	Deleted lemma and POS tag

Table 3: Full feature set in the association model. $x_{i:j}$ and $c_{i':j'}$ denote spans from x and c. pos($x_{i:j}$) and lemma($x_{i:j}$) denote the POS tag and lemma sequence of $x_{i:j}$.

corpora, containing 18 million pairs of question paraphrases from `wikianswers.com`, which were tagged as having the same meaning by users. PARALEX is suitable for our needs since it focuses on *question* paraphrases. For example, the phrase *"do for a living"* occurs mostly in questions, and we can extract associations for this phrase from PARALEX. Paraphrase pairs in PARALEX are word-aligned using standard machine translation methods. We use the word alignments to construct a phrase table by applying the consistent phrase pair heuristic (Och and Ney, 2004) to all 5-grams. This results in a phrase table with approximately 1.3 million phrase pairs. We let $\mathcal{A}$ denote this set of mined candidate associations.

For a pair (x, c), we also consider as candidate associations the set $\mathcal{B}$ (represented implicitly), which contains token pairs $(x_i, c_{i'})$ such that x_i and $c_{i'}$ share the same lemma, the same POS tag, or are linked through a *derivation* link on WordNet (Fellbaum, 1998). This allows us to learn paraphrases for words that appear in our datasets but are not covered by the phrase table, and to handle nominalizations for phrase pairs such as *"Who designed the game of life?"* and *"What game designer is the designer of the game of life?"*.

Our model goes over all possible spans of x and c and constructs all possible associations from $\mathcal{A}$ and $\mathcal{B}$. This results in many poor associations (e.g., *"play"* and *"the"*), but as illustrated in Figure 3, we learn weights that discriminate good from bad associations. Table 3 specifies the full set of features. Note that unlike standard paraphrase detection and RTE systems, we use lexicalized features, firing approximately 400,000 features on WEBQUESTIONS. By extracting POS features, we obtain soft syntactic rules, e.g., the feature "`JJ N` $\wedge$ `N`" indicates that omitting adjectives before nouns is possible. Once associations are constructed, we mark tokens in x and c that were not part of any association, and extract deletion features for their lemmas and POS tags. Thus, we learn that deleting pronouns is acceptable, while deleting nouns is not.

To summarize, the association model links phrases of two utterances in multiple overlapping ways. During training, the model learns which associations are characteristic of paraphrases and which are not.

5.2 Vector space model

The association model relies on having a good set of candidate associations, but mining associations suffers from coverage issues. We now introduce a vector space (VS) model, which assigns a vector representation for each utterance, and learns a scoring function that ranks paraphrase candidates.

We start by constructing vector representations of words. We run the WORD2VEC tool (Mikolov et al., 2013) on lower-cased Wikipedia text (1.59 billion tokens), using the CBOW model with a window of 5 and hierarchical softmax. We also experiment with publicly released word embeddings (Huang et al., 2012), which were trained using both local and global context. Both result in k-dimensional vectors ($k = 50$). Next, we construct a vector $v_x \in \mathbb{R}^k$ for each utterance x by simply averaging the vectors of all content words (nouns, verbs, and adjectives) in x.

We can now estimate a paraphrase score for two utterances x and c via a weighted combination of the components of the vector representations:

$$v_x^\top W v_c = \sum_{i,j=1}^{k} w_{ij} v_{x,i} v_{c,j}$$

where $W \in \mathbb{R}^{k \times k}$ is a parameter matrix. In terms of our earlier notation, we have $\theta_{\text{vs}} = \text{vec}(W)$ and $\phi_{\text{vs}}(x, c) = \text{vec}(v_x v_c^\top)$, where vec$(\cdot)$ unrolls a matrix into a vector. In Section 6, we experiment with W equal to the identity matrix, constraining W to be diagonal, and learning a full W matrix.

The VS model can identify correct paraphrases in cases where it is hard to directly associate phrases from x and c. For example, the answer to *"Where is made Kia car?"* (from WEBQUESTIONS), is given by the canonical utterance *"What city is Kia motors a headquarters of?"*. The association model does not associate *"made"* and *"headquarters"*, but the VS model is able to determine that these utterances are semantically related. In other cases, the VS model cannot distinguish correct paraphrases from incorrect ones. For

Dataset	# examples	# word types
FREE917	917	2,036
WEBQUESTIONS	5,810	4,525

Table 4: Statistics on WEBQUESTIONS and FREE917.

example, the association model identifies that the paraphrase for *"What type of music did Richard Wagner Play?"* is *"What is the musical genres of Richard Wagner?"*, by relating phrases such as *"type of music"* and *"musical genres"*. The VS model ranks the canonical utterance *"What composition has Richard Wagner as lyricist?"* higher, as this utterance is also in the music domain. Thus, we combine the two models to benefit from their complementary nature.

In summary, while the association model aligns particular phrases to one another, the vector space model provides a soft vector-based representation for utterances.

6 Empirical evaluation

In this section, we evaluate our system on WEBQUESTIONS and FREE917. After describing the setup (Section 6.1), we present our main empirical results and analyze the components of the system (Section 6.2).

6.1 Setup

We use the WEBQUESTIONS dataset (Berant et al., 2013), which contains 5,810 question-answer pairs. This dataset was created by crawling questions through the Google Suggest API, and then obtaining answers using Amazon Mechanical Turk. We use the original train-test split, and divide the training set into 3 random 80%–20% splits for development. This dataset is characterized by questions that are commonly asked on the web (and are not necessarily grammatical), such as *"What character did Natalie Portman play in Star Wars?"* and *"What kind of money to take to Bahamas?"*.

The FREE917 dataset contains 917 questions, authored by two annotators and annotated with logical forms. This dataset contains questions on rarer topics (for example, *"What is the engine in a 2010 Ferrari California?"* and *"What was the cover price of the X-men Issue 1?"*), but the phrasing of questions tends to be more rigid compared to WEBQUESTIONS. Table 4 provides some statistics on the two datasets. Following Cai and Yates (2013), we hold out 30% of the data for the final test, and perform 3 random 80%-20% splits of the training set for development. Since we train from question-answer pairs, we collect answers by executing the gold logical forms against Freebase.

We execute λ-DCS queries by converting them into SPARQL and executing them against a copy of Freebase using the Virtuoso database engine. We evaluate our system with accuracy, that is, the proportion of questions we answer correctly. We run all questions through the Stanford CoreNLP pipeline (Toutanova and Manning, 2003; Finkel et al., 2005; Klein and Manning, 2003).

We tuned the L_1 regularization strength, developed features, and ran analysis experiments on the development set (averaging across random splits). On WEBQUESTIONS, without L_1 regularization, the number of non-zero features was 360K; L_1 regularization brings it down to 17K.

6.2 Results

We compare our system to Cai and Yates (2013) (CY13), Berant et al. (2013) (BCFL13), and Kwiatkowski et al. (2013) (KCAZ13). For BCFL13, we obtained results using the SEMPRE package[2] and running Berant et al. (2013)'s system on the datasets.

Table 5 presents results on the test set. We achieve a substantial relative improvement of 12% in accuracy on WEBQUESTIONS, and match the best results on FREE917. Interestingly, our system gets an *oracle accuracy* of 63% on WEBQUESTIONS compared to 48% obtained by BCFL13, where the oracle accuracy is the fraction of questions for which at least one logical form in the candidate set produced by the system is correct. This demonstrates that our method for constructing candidate logical forms is reasonable. To further examine this, we ran BCFL13 on the development set, allowing it to use only predicates from logical forms suggested by our logical form construction step. This improved oracle accuracy on the development set to 64.5%, but accuracy was 32.2%. This shows that the improvement in accuracy should not be attributed only to better logical form generation, but also to the paraphrase model.

We now perform more extensive analysis of our system's components and compare it to various baselines.

Component ablation We ablate the association model, the VS model, and the entire paraphrase

[2] http://www-nlp.stanford.edu/software/sempre/

	FREE917	WEBQUESTIONS
CY13	59.0	–
BCFL13	62.0	35.7
KCAZ13	68.0	–
This work	**68.5**	**39.9**

Table 5: Results on the test set.

	FREE917	WEBQUESTIONS
Our system	**73.9**	**41.2**
–VSM	71.0	40.5
–ASSOCIATION	52.7	35.3
–PARAPHRASE	31.8	21.3
SIMPLEGEN	73.4	40.4
Full matrix	52.7	35.3
Diagonal	50.4	30.6
Identity	50.7	30.4
JACCARD	69.7	31.3
EDIT	40.8	24.8
WDDC06	71.0	29.8

Table 6: Results for ablations and baselines on development set.

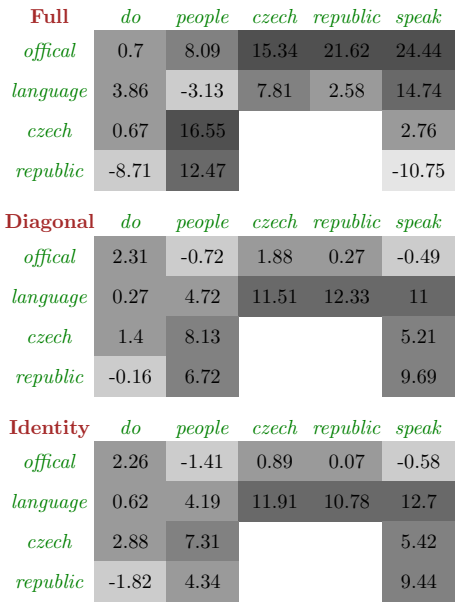


Figure 4: Values of the paraphrase score $v_{x_i}^\top W v_{c_{i'}}$ for all content word tokens x_i and $c_{i'}$, where W is an arbitrary full matrix, a diagonal matrix, or the identity matrix. We omit scores for the words *"czech"* and *"republic"* since they appear in all canonical utterances for this example.

model (using only logical form features). Table 5 shows that our full system obtains highest accuracy, and that removing the association model results in a much larger degradation compared to removing the VS model.

Utterance generation Our system generates relatively natural utterances from logical forms using simple rules based on Freebase descriptions (Section 4). We now consider simply concatenating Freebase descriptions. For example, the logical form `R[PlaceOfBirth].ElvisPresley` would generate the utterance *"What location Elvis Presley place of birth?"*. Row SIMPLEGEN in Table 6 demonstrates that we still get good results in this setup. This is expected given that our paraphrase models are not sensitive to the syntactic structure of the generated utterance.

VS model Our system learns parameters for a full W matrix. We now examine results when learning parameters for a full matrix W, a diagonal matrix W, and when setting W to be the identity matrix. Table 6 (third section) illustrates that learning a full matrix substantially improves accuracy. Figure 4 gives an example for a correct paraphrase pair, where the full matrix model boosts the overall model score. Note that the full matrix assigns a high score for the phrases *"official language"* and *"speak"* compared to the simpler models, but other pairs are less interpretable.

Baselines We also compared our system to the following implemented baselines:

- JACCARD: We compute the Jaccard score between the tokens of x and c and define $\phi_{\text{pr}}(x, c)$ to be this single feature.
- EDIT: We compute the token edit distance between x and c and define $\phi_{\text{pr}}(x, c)$ to be this single feature.
- WDDC06: We re-implement 13 features from Wan et al. (2006), who obtained close to state-of-the-art performance on the Microsoft Research paraphrase corpus.[3]

Table 6 demonstrates that we improve performance over all baselines. Interestingly, JACCARD and WDDC06 obtain reasonable performance on FREE917 but perform much worse on WEBQUESTIONS. We surmise this is because questions in FREE917 were generated by annotators prompted by Freebase facts, whereas questions in WEBQUESTIONS originated independently of Freebase. Thus, word choice in FREE917 is often close to the generated Freebase descriptions, allowing simple baselines to perform well.

Error analysis We sampled examples from the development set to examine the main reasons PARASEMPRE makes errors. We notice that in many cases the paraphrase model can be further improved. For example, PARASEMPRE suggests

[3]We implement all features that do not require dependency parsing.

that the best paraphrase for *"What company did Henry Ford work for?"* is *"What written work novel by Henry Ford?"* rather than *"The employer of Henry Ford"*, due to the exact match of the word *"work"*. Another example is the question *"Where is the Nascar hall of fame?"*, where PARASEMPRE suggests that *"What hall of fame discipline has Nascar hall of fame as halls of fame?"* is the best canonical utterance. This is because our simple model allows to associate *"hall of fame"* with the canonical utterance three times. Entity recognition also accounts for many errors, e.g., the entity chosen in *"where was the gallipoli campaign waged?"* is `Galipoli` and not `GalipoliCampaign`. Last, PARASEMPRE does not handle temporal information, which causes errors in questions like *"Where did Harriet Tubman live after the civil war?"*

7 Discussion

In this work, we approach the problem of semantic parsing from a paraphrasing viewpoint. A fundamental motivation and long standing goal of the paraphrasing and RTE communities has been to cast various semantic applications as paraphrasing/textual entailment (Dagan et al., 2013). While it has been shown that paraphrasing methods are useful for question answering (Harabagiu and Hickl, 2006) and relation extraction (Romano et al., 2006), this is, to the best of our knowledge, the first paper to perform semantic parsing through paraphrasing. Our paraphrase model emphasizes simplicity and efficiency, but the framework is agnostic to the internals of the paraphrase method.

On the semantic parsing side, our work is most related to Kwiatkowski et al. (2013). The main challenge in semantic parsing is coping with the *mismatch* between language and the KB. In both Kwiatkowski et al. (2013) and this work, an intermediate representation is employed to handle the mismatch, but while they use a logical representation, we opt for a text-based one. Our choice allows us to benefit from the parallel monolingual corpus PARALEX and from word vectors trained on Wikipedia. We believe that our approach is particularly suitable for scenarios such as factoid question answering, where the space of logical forms is somewhat constrained and a few generation rules suffice to reduce the problem to paraphrasing.

Our work is also related to Fader et al. (2013), who presented a paraphrase-driven question answering system. One can view this work as a generalization of Fader et al. along three dimensions. First, Fader et al. use a KB over natural language extractions rather than a formal KB and so querying the KB does not require a generation step – they paraphrase questions to KB entries directly. Second, they suggest a particular paraphrasing method that maps a test question to a question for which the answer is already known in a single step. We propose a general paraphrasing framework and instantiate it with two paraphrase models. Lastly, Fader et al. handle queries with only one property and entity whereas we generalize to more types of logical forms.

Since our generated questions are passed to a paraphrase model, we took a very simple approach, mostly ensuring that we preserved the semantics of the utterance without striving for the most fluent realization. Research on generation (Dale et al., 2003; Reiter et al., 2005; Turner et al., 2009; Piwek and Boyer, 2012) typically focuses on generating natural utterances for human consumption, where fluency is important.

In conclusion, the main contribution of this paper is a novel approach for semantic parsing based on a simple generation procedure and a paraphrase model. We achieve state-of-the-art results on two recently released datasets. We believe that our approach opens a window of opportunity for learning semantic parsers from raw text not necessarily related to the target KB. With more sophisticated generation and paraphrase, we hope to tackle compositionally richer utterances.

Acknowledgments

We thank Kai Sheng Tai for performing the error analysis. Stanford University gratefully acknowledges the support of the Defense Advanced Research Projects Agency (DARPA) Deep Exploration and Filtering of Text (DEFT) Program under Air Force Research Laboratory (AFRL) contract no. FA8750-13-2-0040. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, AFRL, or the US government. The second author is supported by a Google Faculty Research Award.

References

J. Berant, A. Chou, R. Frostig, and P. Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Q. Cai and A. Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *Association for Computational Linguistics (ACL)*.

M. Chang, D. Goldwasser, D. Roth, and V. Srikumar. 2010. Discriminative learning over constrained latent representations. In *North American Association for Computational Linguistics (NAACL)*.

I. Dagan, D. Roth, M. Sammons, and F. M. Zanzotto. 2013. *Recognizing Textual Entailment: Models and Applications*. Morgan and Claypool Publishers.

R. Dale, S. Geldof, and J. Prost. 2003. Coral: using natural language generation for navigational assistance. In *Australasian computer science conference*, pages 35–44.

D. Das and N. A. Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Association for Computational Linguistics (ACL)*, pages 468–476.

B. Dolan, C. Quirk, and C. Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *International Conference on Computational Linguistics (COLING)*.

J. Duchi, E. Hazan, and Y. Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. In *Conference on Learning Theory (COLT)*.

A. Fader, S. Soderland, and O. Etzioni. 2011. Identifying relations for open information extraction. In *Empirical Methods in Natural Language Processing (EMNLP)*.

A. Fader, L. Zettlemoyer, and O. Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Association for Computational Linguistics (ACL)*.

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

J. R. Finkel, T. Grenager, and C. Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Association for Computational Linguistics (ACL)*, pages 363–370.

Google. 2013. Freebase data dumps (2013-06-09). `https://developers.google.com/freebase/data`.

A. Haghighi, A. Y. Ng, and C. D. Manning. 2005. Robust textual inference via graph matching. In *Empirical Methods in Natural Language Processing (EMNLP)*.

S. Harabagiu and A. Hickl. 2006. Methods for using textual entailment in open-domain question answering. In *Association for Computational Linguistics (ACL)*.

M. Heilman and N. A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Human Language Technology and North American Association for Computational Linguistics (HLT/NAACL)*, pages 1011–1019.

E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Association for Computational Linguistics (ACL)*.

D. Klein and C. Manning. 2003. Accurate unlexicalized parsing. In *Association for Computational Linguistics (ACL)*, pages 423–430.

T. Kwiatkowski, L. Zettlemoyer, S. Goldwater, and M. Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1223–1233.

T. Kwiatkowski, E. Choi, Y. Artzi, and L. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Empirical Methods in Natural Language Processing (EMNLP)*.

P. Liang. 2013. Lambda dependency-based compositional semantics. Technical report, ArXiv.

T. Lin, Mausam, and O. Etzioni. 2012. Entity linking at web scale. In *Knowledge Extraction Workshop (AKBC-WEKEX)*.

T. Mikolov, K. Chen, G. Corrado, and Jeffrey. 2013. Efficient estimation of word representations in vector space. Technical report, ArXiv.

F. J. Och and H. Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30:417–449.

P. Piwek and K. E. Boyer. 2012. Varieties of question generation: Introduction to this special issue. *Dialogue and Discourse*, 3:1–9.

E. Reiter, S. Sripada, J. Hunter, J. Yu, and I. Davy. 2005. Choosing words in computer-generated weather forecasts. *Artificial Intelligence*, 167:137–169.

L. Romano, M. kouylekov, I. Szpektor, I. Dagan, and A. Lavelli. 2006. Investigating a generic paraphrase-based approach for relation extraction. In *Proceedings of ECAL*.

R. Socher, E. H. Huang, J. Pennin, C. D. Manning, and A. Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems (NIPS)*, pages 801–809.

A. Stern and I. Dagan. 2011. A confidence model for syntactically-motivated entailment proofs. In *Recent Advances in Natural Language Processing*, pages 455–462.

K. Toutanova and C. D. Manning. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Human Language Technology and North American Association for Computational Linguistics (HLT/NAACL)*.

R. Turner, Y. Sripada, and E. Reiter. 2009. Generating approximate geographic descriptions. In *European Workshop on Natural Language Generation*, pages 42–49.

S. Wan, M. Dras, R. Dale, and C. Paris. 2006. Using dependency-based features to take the "para-farce" out of paraphrase. In *Australasian Language Technology Workshop*.

M. Wang and C. D. Manning. 2010. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *The International Conference on Computational Linguistics*, pages 1164–1172.

Y. W. Wong and R. J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Association for Computational Linguistics (ACL)*, pages 960–967.

M. Zelle and R. J. Mooney. 1996. Learning to parse database queries using inductive logic proramming. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 1050–1055.

L. S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Uncertainty in Artificial Intelligence (UAI)*, pages 658–666.

A Discriminative Graph-Based Parser for the Abstract Meaning Representation

Jeffrey Flanigan Sam Thomson Jaime Carbonell Chris Dyer Noah A. Smith
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{jflanigan,sthomson,jgc,cdyer,nasmith}@cs.cmu.edu

Abstract

Abstract Meaning Representation (AMR) is a semantic formalism for which a growing set of annotated examples is available. We introduce the first approach to parse sentences into this representation, providing a strong baseline for future improvement. The method is based on a novel algorithm for finding a maximum spanning, connected subgraph, embedded within a Lagrangian relaxation of an optimization problem that imposes linguistically inspired constraints. Our approach is described in the general framework of structured prediction, allowing future incorporation of additional features and constraints, and may extend to other formalisms as well. Our open-source system, JAMR, is available at:
http://github.com/jflanigan/jamr

1 Introduction

Semantic parsing is the problem of mapping natural language strings into meaning representations. Abstract Meaning Representation (AMR) (Banarescu et al., 2013; Dorr et al., 1998) is a semantic formalism in which the meaning of a sentence is encoded as a rooted, directed, acyclic graph. Nodes represent concepts, and labeled directed edges represent the relationships between them–see Figure 1 for an example AMR graph. The formalism is based on propositional logic and neo-Davidsonian event representations (Parsons, 1990; Davidson, 1967). Although it does not encode quantifiers, tense, or modality, the set of semantic phenomena included in AMR were selected with natural language applications—in particular, machine translation—in mind.

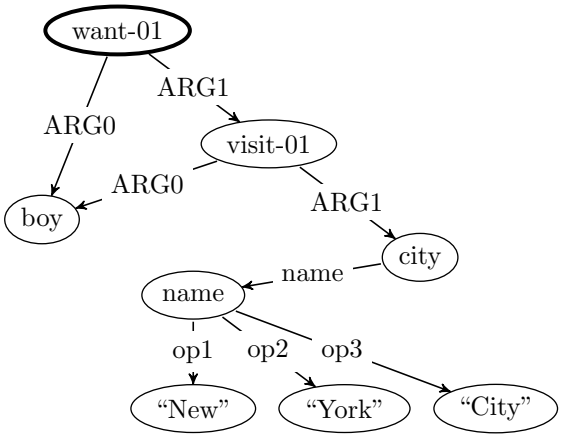
In this paper we introduce JAMR, the first published system for automatic AMR parsing. The system is based on a statistical model whose parameters are trained discriminatively using annotated sentences in the AMR Bank corpus (Banarescu et al., 2013). We evaluate using the Smatch score (Cai and Knight, 2013), establishing a baseline for future work.

The core of JAMR is a two-part algorithm that first identifies *concepts* using a semi-Markov model and then identifies the *relations* that obtain between these by searching for the **maximum spanning connected subgraph** (MSCG) from an edge-labeled, directed graph representing all possible relations between the identified concepts. To solve the latter problem, we introduce an apparently novel $O(|V|^2 \log |V|)$ algorithm that is similar to the maximum spanning tree (MST) algorithms that are widely used for dependency parsing (McDonald et al., 2005). Our MSCG algorithm returns the connected subgraph with maximal sum of its edge weights from among all connected subgraphs of the input graph. Since AMR imposes additional constraints to ensure semantic well-formedness, we use Lagrangian relaxation (Geoffrion, 1974; Fisher, 2004) to augment the MSCG algorithm, yielding a tractable iterative algorithm that finds the optimal solution subject to these constraints. In our experiments, we have found this algorithm to converge 100% of the time for the constraint set we use.

The approach can be understood as an alternative to parsing approaches using graph transducers such as (synchronous) hyperedge replacement grammars (Chiang et al., 2013; Jones et al., 2012; Drewes et al., 1997), in much the same way that spanning tree algorithms are an alternative to using shift-reduce and dynamic programming algorithms for dependency parsing.[1] While a detailed

[1] To date, a graph transducer-based semantic parser has not been published, although the Bolinas toolkit (http://www.isi.edu/publications/licensed-sw/bolinas/) contains much of the necessary infrastructure.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1426–1436,
Baltimore, Maryland, USA, June 23-25 2014.



(a) Graph.

```
(w / want-01
  :ARG0 (b / boy)
  :ARG1 (g / visit-01
           :ARG0 b
           :ARG1 (c / city
                    :name (n / name
                             :op1 "New"
                             :op2 "York"
                             :op3 "City"))))
```

(b) AMR annotation.

Figure 1: Two equivalent ways of representing the AMR parse for the sentence, "The boy wants to visit New York City."

comparison of these two approaches is beyond the scope of this paper, we emphasize that—as has been observed with dependency parsing—a diversity of approaches can shed light on complex problems such as semantic parsing.

2 Notation and Overview

Our approach to AMR parsing represents an AMR parse as a graph $G = \langle V, E \rangle$; vertices and edges are given labels from sets L_V and L_E, respectively. G is constructed in two stages. The first stage identifies the **concepts** evoked by words and phrases in an input sentence $\boldsymbol{w} = \langle w_1, \ldots, w_n \rangle$, each w_i a member of vocabulary W. The second stage connects the concepts by adding L_E-labeled edges capturing the **relations** between concepts, and selects a root in G corresponding to the **focus** of the sentence $\boldsymbol{w}$.

Concept identification (§3) involves segmenting $\boldsymbol{w}$ into contiguous spans and assigning to each span a graph fragment corresponding to a concept from a concept set denoted F (or to $\emptyset$ for words that evoke no concept). In §5 we describe how F is constructed. In our formulation, spans are contiguous subsequences of $\boldsymbol{w}$. For example, the words "New York City" can evoke the fragment represented by

```
(c / city
  :name (n / name
           :op1 "New"
           :op2 "York"
           :op3 "City"))))
```

We use a sequence labeling algorithm to identify concepts.

The relation identification stage (§4) is similar to a graph-based dependency parser. Instead of finding the maximum-scoring tree over words, it finds the maximum-scoring connected subgraph that preserves concept fragments from the first stage, links each pair of vertices by at most one edge, and is deterministic[2] with respect to a special set of edge labels $L_E^* \subset L_E$. The set L_E^* consists of the labels ARG0–ARG5, and does not include labels such as MOD or MANNER, for example. Linguistically, the determinism constraint enforces that predicates have at most one semantic argument of each type; this is discussed in more detail in §4.

To train the parser, spans of words must be labeled with the concept fragments they evoke. Although AMR Bank does not label concepts with the words that evoke them, it is possible to build an automatic aligner (§5). The alignments are used to construct the concept lexicon and to train the concept identification and relation identification stages of the parser (§6). Each stage is a discriminatively-trained linear structured predictor with rich features that make use of part-of-speech tagging, named entity tagging, and dependency parsing.

In §7, we evaluate the parser against gold-standard annotated sentences from the AMR Bank corpus (Banarescu et al., 2013) under the Smatch score (Cai and Knight, 2013), presenting the first published results on automatic AMR parsing.

3 Concept Identification

The concept identification stage maps spans of words in the input sentence $\boldsymbol{w}$ to concept graph fragments from F, or to the empty graph fragment $\emptyset$. These graph fragments often consist of just one labeled concept node, but in some cases they are larger graphs with multiple nodes and edges.[3]

[2] By this we mean that, at each node, there is at most one outgoing edge with that label type.

[3] About 20% of invoked concept fragments are multi-concept fragments.

Concept identification is illustrated in Figure 2 using our running example, "The boy wants to visit New York City."

Let the concept lexicon be a mapping $clex : W^* \rightarrow 2^F$ that provides candidate graph fragments for sequences of words. (The construction of F and $clex$ is discussed below.) Formally, a concept labeling is (i) a segmentation of $\boldsymbol{w}$ into contiguous spans represented by boundaries $\boldsymbol{b}$, giving spans $\langle \boldsymbol{w}_{b_0:b_1}, \boldsymbol{w}_{b_1:b_2}, \ldots \boldsymbol{w}_{b_{k-1}:b_k} \rangle$, with $b_0 = 0$ and $b_k = n$, and (ii) an assignment of each phrase $\boldsymbol{w}_{b_{i-1}:b_i}$ to a concept graph fragment $c_i \in clex(\boldsymbol{w}_{b_{i-1}:b_i}) \cup \emptyset$.

Our approach scores a sequence of spans $\boldsymbol{b}$ and a sequence of concept graph fragments $\boldsymbol{c}$, both of arbitrary length k, using the following locally decomposed, linearly parameterized function:

$$score(\boldsymbol{b}, \boldsymbol{c}; \boldsymbol{\theta}) = \sum_{i=1}^{k} \boldsymbol{\theta}^\top \mathbf{f}(\boldsymbol{w}_{b_{i-1}:b_i}, b_{i-1}, b_i, c_i) \quad (1)$$

where $\mathbf{f}$ is a feature vector representation of a span and one of its concept graph fragments in context. The features are:

- **Fragment given words**: Relative frequency estimates of the probability of a concept graph fragment given the sequence of words in the span. This is calculated from the concept-word alignments in the training corpus (§5).
- **Length** of the matching span (number of tokens).
- **NER**: 1 if the named entity tagger marked the span as an entity, 0 otherwise.
- **Bias**: 1 for any concept graph fragment from F and 0 for $\emptyset$.

Our approach finds the highest-scoring $\boldsymbol{b}$ and $\boldsymbol{c}$ using a dynamic programming algorithm: the zeroth-order case of inference under a semi-Markov model (Janssen and Limnios, 1999). Let $S(i)$ denote the score of the best labeling of the first i words of the sentence, $\boldsymbol{w}_{0:i}$; it can be calculated using the recurrence:

$$S(0) = 0$$
$$S(i) = \max_{\substack{j:0 \leq j < i, \\ c \in clex(\boldsymbol{w}_{j:i}) \cup \emptyset}} \left\{ S(j) + \boldsymbol{\theta}^\top \mathbf{f}(\boldsymbol{w}_{j:i}, j, i, c) \right\}$$

The best score will be $S(n)$, and the best scoring concept labeling can be recovered using backpointers, as in typical implementations of the Viterbi algorithm. Runtime is $O(n^2)$.

$clex$ is implemented as follows. When $clex$ is called with a sequence of words, it looks up the sequence in a table that contains, for every word sequence that was labeled with a concept fragment in the training data, the set of concept fragments it was labeled with. $clex$ also has a set of rules for generating concept fragments for named entities and time expressions. It generates a concept fragment for any entity recognized by the named entity tagger, as well as for any word sequence matching a regular expression for a time expression. $clex$ returns the union of all these concept fragments.

4 Relation Identification

The relation identification stage adds edges among the concept subgraph fragments identified in the first stage (§3), creating a graph. We frame the task as a constrained combinatorial optimization problem.

Consider the fully dense labeled multigraph $D = \langle V_D, E_D \rangle$ that includes the union of all labeled vertices and labeled edges in the concept graph fragments, as well as every possible labeled edge $u \xrightarrow{\ell} v$, for all $u, v \in V_D$ and every $\ell \in L_E$.[4]

We require a subgraph $G = \langle V_G, E_G \rangle$ that respects the following constraints:

1. **Preserving**: all graph fragments (including labels) from the concept identification phase are subgraphs of G.
2. **Simple**: for any two vertices u and $v \in V_G$, E_G includes at most one edge between u and v. This constraint forbids a small number of perfectly valid graphs, for example for sentences such as "John hurt himself"; however, we see that $< 1\%$ of training instances violate the constraint. We found in preliminary experiments that including the constraint increases overall performance.[5]
3. **Connected**: G must be weakly connected (every vertex reachable from every other vertex, ignoring the direction of edges). This constraint follows from the formal definition of AMR and is never violated in the training data.
4. **Deterministic**: For each node $u \in V_G$, and for each label $\ell \in L_E^*$, there is at most one outgoing edge in E_G from u with label ℓ. As discussed in §2, this constraint is linguistically motivated.

[4] To handle numbered OP labels, we pre-process the training data to convert OPN to OP, and post-process the output by numbering the OP labels sequentially.

[5] In future work it might be treated as a soft constraint, or the constraint might be refined to specific cases.

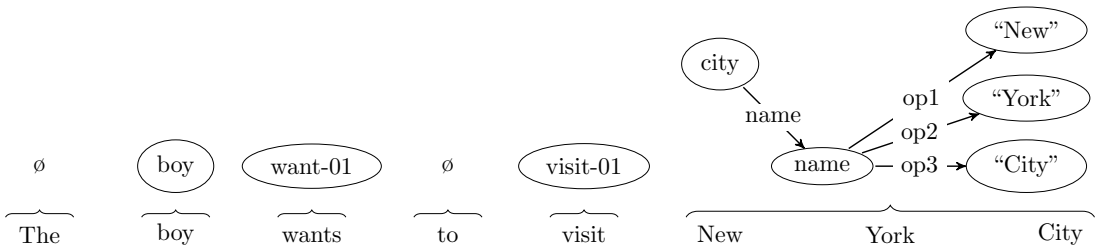


Figure 2: A concept labeling for the sentence "The boy wants to visit New York City."

One constraint we do not include is **acyclicity**, which follows from the definition of AMR. In practice, graphs with cycles are rarely produced by JAMR. In fact, none of the graphs produced on the test set violate acyclicity.

Given the constraints, we seek the maximum-scoring subgraph. We define the score to decompose by edges, and with a linear parameterization:

$$score(E_G; \boldsymbol{\psi}) = \sum_{e \in E_G} \boldsymbol{\psi}^\top \mathbf{g}(e) \qquad (2)$$

The features are shown in Table 1.

Our solution to maximizing the score in Eq. 2, subject to the constraints, makes use of (i) an algorithm that ignores constraint 4 but respects the others (§4.1); and (ii) a Lagrangian relaxation that iteratively adjusts the edge scores supplied to (i) so as to enforce constraint 4 (§4.2).

4.1 Maximum Preserving, Simple, Spanning, Connected Subgraph Algorithm

The steps for constructing a maximum preserving, simple, spanning, connected (but not necessarily deterministic) subgraph are as follows. These steps ensure the resulting graph G satisfies the constraints: the initialization step ensures the preserving constraint is satisfied, the pre-processing step ensures the graph is simple, and the core algorithm ensures the graph is connected.

1. (Initialization) Let $E^{(0)}$ be the union of the concept graph fragments' weighted, labeled, directed edges. Let V denote its set of vertices. Note that $\langle V, E^{(0)} \rangle$ is preserving (constraint 4), as is any graph that contains it. It is also simple (constraint 4), assuming each concept graph fragment is simple.
2. (Pre-processing) We form the edge set E by including just one edge from E_D between each pair of nodes:
   - For any edge $e = u \xrightarrow{\ell} v$ in $E^{(0)}$, include e in E, omitting all other edges between u and v.
   - For any two nodes u and v, include only the highest scoring edge between u and v.

   Note that without the deterministic constraint, we have no constraints that depend on the label of an edge, nor its direction. So it is clear that the edges omitted in this step could not be part of the maximum-scoring solution, as they could be replaced by a higher scoring edge without violating any constraints.

   Note also that because we have kept exactly one edge between every pair of nodes, $\langle V, E \rangle$ is simple and connected.
3. (Core algorithm) Run Algorithm 1, MSCG, on $\langle V, E \rangle$ and $E^{(0)}$. This algorithm is a (to our knowledge novel) modification of the minimum spanning tree algorithm of Kruskal (1956). Note that the directions of edges do not matter for MSCG.

Steps 1–2 can be accomplished in one pass through the edges, with runtime $O(|V|^2)$. MSCG can be implemented efficiently in $O(|V|^2 \log |V|)$ time, similarly to Kruskal's algorithm, using a disjoint-set data structure to keep track of connected components.[6] The total asymptotic runtime complexity is $O(|V|^2 \log |V|)$.

The details of MSCG are given in Algorithm 1. In a nutshell, MSCG first adds all positive edges to the graph, and then connects the graph by greedily adding the least negative edge that connects two previously unconnected components.

Theorem 1. MSCG *finds a maximum spanning, connected subgraph of* $\langle V, E \rangle$

Proof. We closely follow the original proof of correctness of Kruskal's algorithm. We first show by induction that, at every iteration of MSCG, there exists some maximum spanning, connected subgraph that contains $G^{(i)} = \langle V, E^{(i)} \rangle$:

[6] For dense graphs, Prim's algorithm (Prim, 1957) is asymptotically faster ($O(|V|^2)$). We conjecture that using Prim's algorithm instead of Kruskall's to connect the graph could improve the runtime of MSCG.

Name	Description
Label	For each $\ell \in L_E$, 1 if the edge has that label
Self edge	1 if the edge is between two nodes in the same fragment
Tail fragment root	1 if the edge's tail is the root of its graph fragment
Head fragment root	1 if the edge's head is the root of its graph fragment
Path	Dependency edge labels and parts of speech on the shortest syntactic path between any two words in the two spans
Distance	Number of tokens (plus one) between the two concepts' spans (zero if the same)
Distance indicators	A feature for each distance value, that is 1 if the spans are of that distance
Log distance	Logarithm of the distance feature plus one.
Bias	1 for any edge.

Table 1: Features used in relation identification. In addition to the features above, the following conjunctions are used (Tail and Head concepts are elements of L_V): Tail concept ∧ Label, Head concept ∧ Label, Path ∧ Label, Path ∧ Head concept, Path ∧ Tail concept, Path ∧ Head concept ∧ Label, Path ∧ Tail concept ∧ Label, Path ∧ Head word, Path ∧ Tail word, Path ∧ Head word ∧ Label, Path ∧ Tail word ∧ Label, Distance ∧ Label, Distance ∧ Path, and Distance ∧ Path ∧ Label. To conjoin the distance feature with anything else, we multiply by the distance.

input : weighted, connected graph $\langle V, E\rangle$ and set of edges $E^{(0)} \subseteq E$ to be preserved
output: maximum spanning, connected subgraph of $\langle V, E\rangle$ that preserves $E^{(0)}$

let $E^{(1)} = E^{(0)} \cup \{e \in E \mid \boldsymbol{\psi}^\top \mathbf{g}(e) > 0\}$;
create a priority queue Q containing $\{e \in E \mid \boldsymbol{\psi}^\top \mathbf{g}(e) \leq 0\}$ prioritized by scores;
$i = 1$;
while *Q nonempty and $\langle V, E^{(i)}\rangle$ is not yet spanning and connected* **do**
  $i = i + 1$;
  $E^{(i)} = E^{(i-1)}$;
  $e = \arg\max_{e' \in Q} \boldsymbol{\psi}^\top \mathbf{g}(e')$;
  remove e from Q;
  if *e connects two previously unconnected components of $\langle V, E^{(i)}\rangle$* **then**
    add e to $E^{(i)}$
  end
end
return $G = \langle V, E^{(i)}\rangle$;

Algorithm 1: MSCG algorithm.

Base case: Consider $G^{(1)}$, the subgraph containing $E^{(0)}$ and every positive edge. Take any maximum preserving spanning connected subgraph M of $\langle V, E\rangle$. We know that such an M exists because $\langle V, E\rangle$ itself is a preserving spanning connected subgraph. Adding a positive edge to M would strictly increase M's score without disconnecting M, which would contradict the fact that M is maximal. Thus M must contain $G^{(1)}$.

Induction step: By the inductive hypothesis, there exists some maximum spanning connected subgraph $M = \langle V, E_M\rangle$ that contains $G^{(i)}$.

Let e be the next edge added to $E^{(i)}$ by MSCG.

If e is in E_M, then $E^{(i+1)} = E^{(i)} \cup \{e\} \subseteq E_M$, and the hypothesis still holds.

Otherwise, since M is connected and does not contain e, $E_M \cup \{e\}$ must have a cycle containing e. In addition, that cycle must have some edge e' that is not in $E^{(i)}$. Otherwise, $E^{(i)} \cup \{e\}$ would contain a cycle, and e would not connect two unconnected components of $G^{(i)}$, contradicting the fact that e was chosen by MSCG.

Since e' is in a cycle in $E_M \cup \{e\}$, removing it will not disconnect the subgraph, i.e. $(E_M \cup \{e\}) \setminus \{e'\}$ is still connected and spanning. The score of e is greater than or equal to the score of e', otherwise MSCG would have chosen e' instead of e. Thus, $\langle V, (E_M \cup \{e\}) \setminus \{e'\}\rangle$ is a maximum spanning connected subgraph that contains $E^{(i+1)}$, and the hypothesis still holds.

When the algorithm completes, $G = \langle V, E^{(i)}\rangle$ is a spanning connected subgraph. The maximum spanning connected subgraph M that contains it cannot have a higher score, because G contains every positive edge. Hence G is maximal. □

4.2 Lagrangian Relaxation

If the subgraph resulting from MSCG satisfies constraint 4 (deterministic) then we are done. Otherwise we resort to Lagrangian relaxation (LR). Here we describe the technique as it applies to our task, referring the interested reader to Rush and Collins (2012) for a more general introduction to Lagrangian relaxation in the context of structured prediction problems.

In our case, we begin by encoding a graph $G = \langle V_G, E_G\rangle$ as a binary vector. For each edge e in the fully dense multigraph D, we associate a bi-

nary variable $z_e = \mathbf{1}\{e \in E_G\}$, where $\mathbf{1}\{P\}$ is the indicator function, taking value 1 if the proposition P is true, 0 otherwise. The collection of z_e form a vector $\mathbf{z} \in \{0,1\}^{|E_D|}$.

Determinism constraints can be encoded as a set of linear inequalities. For example, the constraint that vertex u has no more than one outgoing ARG0 can be encoded with the inequality:

$$\sum_{v \in V} \mathbf{1}\{u \xrightarrow{\text{ARG0}} v \in E_G\} = \sum_{v \in V} z_{u \xrightarrow{\text{ARG0}} v} \leq 1.$$

All of the determinism constraints can collectively be encoded as one system of inequalities:

$$\mathbf{A}\mathbf{z} \leq \mathbf{b},$$

with each row $\mathbf{A}_i$ in A and its corresponding entry b_i in $\mathbf{b}$ together encoding one constraint. For the previous example we have a row $\mathbf{A}_i$ that has 1s in the columns corresponding to edges outgoing from u with label ARG0 and 0's elsewhere, and a corresponding element $b_i = 1$ in $\mathbf{b}$.

The score of graph G (encoded as $\mathbf{z}$) can be written as the objective function $\boldsymbol{\phi}^\top \mathbf{z}$, where $\phi_e = \boldsymbol{\psi}^\top \mathbf{g}(e)$. To handle the constraint $\mathbf{A}\mathbf{z} \leq \mathbf{b}$, we introduce multipliers $\boldsymbol{\mu} \geq 0$ to get the Lagrangian relaxation of the objective function:

$$L_{\boldsymbol{\mu}}(\mathbf{z}) = \max_{\mathbf{z}} (\boldsymbol{\phi}^\top \mathbf{z} + \boldsymbol{\mu}^\top (\mathbf{b} - \mathbf{A}\mathbf{z})),$$

$$\mathbf{z}^*_{\boldsymbol{\mu}} = \arg\max_{\mathbf{z}} L_{\boldsymbol{\mu}}(\mathbf{z}).$$

And the dual objective:

$$L(\mathbf{z}) = \min_{\boldsymbol{\mu} \geq \mathbf{0}} L_{\boldsymbol{\mu}}(\mathbf{z}),$$

$$\mathbf{z}^* = \arg\max_{\mathbf{z}} L(\mathbf{z}).$$

Conveniently, $L_{\boldsymbol{\mu}}(\mathbf{z})$ decomposes over edges:

$$\begin{aligned} L_{\boldsymbol{\mu}}(\mathbf{z}) &= \max_{\mathbf{z}} (\boldsymbol{\phi}^\top \mathbf{z} + \boldsymbol{\mu}^\top (\mathbf{b} - \mathbf{A}\mathbf{z})) \\ &= \max_{\mathbf{z}} (\boldsymbol{\phi}^\top \mathbf{z} - \boldsymbol{\mu}^\top \mathbf{A}\mathbf{z}) \\ &= \max_{\mathbf{z}} ((\boldsymbol{\phi} - \mathbf{A}^\top \boldsymbol{\mu})^\top \mathbf{z}). \end{aligned}$$

So for any $\boldsymbol{\mu}$, we can find $\mathbf{z}^*_\mu$ by assigning edges the new Lagrangian adjusted weights $\boldsymbol{\phi} - \mathbf{A}^\top \boldsymbol{\mu}$ and reapplying the algorithm described in §4.1. We can find $\mathbf{z}^*$ by projected subgradient descent, by starting with $\boldsymbol{\mu} = \mathbf{0}$, and taking steps in the direction:

$$-\frac{\partial L_{\boldsymbol{\mu}}}{\partial \boldsymbol{\mu}}(\mathbf{z}^*_{\boldsymbol{\mu}}) = \mathbf{A}\mathbf{z}^*_{\boldsymbol{\mu}}.$$

If any components of $\boldsymbol{\mu}$ are negative after taking a step, they are set to zero.

$L(\mathbf{z})$ is an upper bound on the unrelaxed objective function $\boldsymbol{\phi}^\top \mathbf{z}$, and is equal to it if and only if the constraints $A\mathbf{z} \leq \mathbf{b}$ are satisfied. If $L(\mathbf{z}^*) = \boldsymbol{\phi}^\top \mathbf{z}^*$, then $\mathbf{z}^*$ is also the optimal solution to the constrained solution. Otherwise, there exists a duality gap, and Lagrangian relaxation has failed. In that case we still return the subgraph encoded by $\mathbf{z}^*$, even though it might violate one or more constraints. Techniques from integer programming such as branch-and-bound or cutting-planes methods could be used to find an optimal solution when LR fails (Das et al., 2012), but we do not use these techniques here. In our experiments, with a stepsize of 1 and max number of steps as 500, Lagrangian relaxation succeeds 100% of the time in our data.

4.3 Focus Identification

In AMR, one node must be marked as the focus of the sentence. We notice this can be accomplished within the relation identification step: we add a special concept node `root` to the dense graph D, and add an edge from `root` to every other node, giving each of these edges the label FOCUS. We require that `root` have at most one outgoing FOCUS edge. Our system has two feature types for this edge: the concept it points to, and the shortest dependency path from a word in the span to the root of the dependency tree.

5 Automatic Alignments

In order to train the parser, we need alignments between sentences in the training data and their annotated AMR graphs. More specifically, we need to know which spans of words invoke which concept fragments in the graph. To do this, we built an automatic aligner and tested its performance on a small set of alignments we annotated by hand.

The automatic aligner uses a set of rules to greedily align concepts to spans. The list of rules is given in Table 2. The aligner proceeds down the list, first aligning named-entities exactly, then fuzzy matching named-entities, then date-entities, etc. For each rule, an entire pass through the AMR graph is done. The pass considers every concept in the graph and attempts to align a concept fragment rooted at that concept if the rule can apply. Some rules only apply to a particular type of concept fragment, while others can apply to any concept. For example, rule 1 can apply to any NAME concept and its OP children. It searches the sentence

for a sequence of words that exactly matches its OP children and aligns them to the NAME and OP children fragment.

Concepts are considered for alignment in the order they are listed in the AMR annotation (left to right, top to bottom). Concepts that are not aligned in a particular pass may be aligned in subsequent passes. Concepts are aligned to the first matching span, and alignments are mutually exclusive. Once aligned, a concept in a fragment is never realigned.[7] However, more concepts can be attached to the fragment by rules 8–14.

We use WordNet to generate candidate lemmas, and we also use a fuzzy match of a concept, defined to be a word in the sentence that has the longest string prefix match with that concept's label, if the match length is ≥ 4. If the match length is < 4, then the concept has no fuzzy match. For example the fuzzy match for ACCUSE-01 could be "accusations" if it is the best match in the sentence. WordNet lemmas and fuzzy matches are only used if the rule explicitly uses them. All tokens and concepts are lowercased before matches or fuzzy matches are done.

On the 200 sentences of training data we aligned by hand, the aligner achieves 92% precision, 89% recall, and 90% F_1 for the alignments.

6 Training

We now describe how to train the two stages of the parser. The training data for the concept identification stage consists of (X, Y) pairs:

- **Input:** X, a sentence annotated with named entities (person, organization, location, misciscellaneous) from the Illinois Named Entity Tagger (Ratinov and Roth, 2009), and part-of-speech tags and basic dependencies from the Stanford Parser (Klein and Manning, 2003; de Marneffe et al., 2006).
- **Output:** Y, the sentence labeled with concept subgraph fragments.

The training data for the relation identification stage consists of (X, Y) pairs:

[7] As an example, if "North Korea" shows up twice in the AMR graph and twice in the input sentence, then the first "North Korea" concept fragment listed in the AMR gets aligned to the first "North Korea" mention in the sentence, and the second fragment to the second mention (because the first span is already aligned when the second "North Korea" concept fragment is considered, so it is aligned to the second matching span).

1. **(Named Entity)** Applies to `name` concepts and their `opn` children. Matches a span that exactly matches its `opn` children in numerical order.

2. **(Fuzzy Named Entity)** Applies to `name` concepts and their `opn` children. Matches a span that matches the fuzzy match of each child in numerical order.

3. **(Date Entity)** Applies to `date-entity` concepts and their `day`, `month`, `year` children (if exist). Matches any permutation of day, month, year, (two digit or four digit years), with or without spaces.

4. **(Minus Polarity Tokens)** Applies to − concepts, and matches "no", "not", "non."

5. **(Single Concept)** Applies to any concept. Strips off trailing '-[0-9]+' from the concept (for example `run-01` → `run`), and matches any exact matching word or WordNet lemma.

6. **(Fuzzy Single Concept)** Applies to any concept. Strips off trailing '-[0-9]+', and matches the fuzzy match of the concept.

7. **(U.S.)** Applies to `name` if its `op1` child is `united` and its `op2` child is `states`. Matches a word that matches "us", "u.s." (no space), or "u. s." (with space).

8. **(Entity Type)** Applies to concepts with an outgoing `name` edge whose head is an aligned fragment. Updates the fragment to include the unaligned concept. Ex: `continent` in `(continent :name (name :op1 "Asia"))` aligned to "asia."

9. **(Quantity)** Applies to `.*-quantity` concepts with an outgoing `unit` edge whose head is aligned. Updates the fragment to include the unaligned concept. Ex: `distance-quantity` in `(distance-quantity :unit kilometer)` aligned to "kilometres."

10. **(Person-Of, Thing-Of)** Applies to `person` and `thing` concepts with an outgoing `.*-of` edge whose head is aligned. Updates the fragment to include the unaligned concept. Ex: `person` in `(person :ARG0-of strike-02)` aligned to "strikers."

11. **(Person)** Applies to `person` concepts with a single outgoing edge whose head is aligned. Updates the fragment to include the unaligned concept. Ex: `person` in `(person :poss (country :name (name :op1 "Korea")))`

12. **(Goverment Organization)** Applies to concepts with an incoming `ARG.*-of` edge whose tail is an aligned `government-organization` concept. Updates the fragment to include the unaligned concept. Ex: `govern-01` in `(government-organization :ARG0-of govern-01)` aligned to "government."

13. **(Minus Polarity Prefixes)** Applies to − concepts with an incoming `polarity` edge whose tail is aligned to a word beginning with "un", "in", or "il." Updates the fragment to include the unaligned concept. Ex: − in `(employ-01 :polarity -)` aligned to "unemployment."

14. **(Degree)** Applies to concepts with an incoming `degree` edge whose tail is aligned to a word ending is "est." Updates the fragment to include the unaligned concept. Ex: `most` in `(large :degree most)` aligned to "largest."

Table 2: Rules used in the automatic aligner.

- **Input:** X, the sentence labeled with graph fragments, as well as named enties, POS tags, and basic dependencies as in concept identification.
- **Output:** Y, the sentence with a full AMR parse.[8]

Alignments are used to induce the concept labeling for the sentences, so no annotation beyond the automatic alignments is necessary.

We train the parameters of the stages separately using AdaGrad (Duchi et al., 2011) with the perceptron loss function (Rosenblatt, 1957; Collins, 2002). We give equations for concept identification parameters $\boldsymbol{\theta}$ and features $\mathbf{f}(X, Y)$. For a sentence of length k, and spans $\boldsymbol{b}$ labeled with a sequence of concept fragments $\boldsymbol{c}$, the features are:

$$\mathbf{f}(X, Y) = \sum_{i=1}^{k} \mathbf{f}(\boldsymbol{w}_{b_{i-1}:b_i}, b_{i-1}, b_i, c_i)$$

To train with AdaGrad, we process examples in the training data $((X^1, Y^1), \ldots, (X^N, Y^N))$ one at a time. At time t, we decode (§3) to get $\hat{Y}^t$ and compute the subgradient:

$$\mathbf{s}^t = \mathbf{f}(X^t, \hat{Y}^t) - \mathbf{f}(X^t, Y^t)$$

We then update the parameters and go to the next example. Each component i of the parameter vector gets updated like so:

$$\theta_i^{t+1} = \theta_i^t - \frac{\eta}{\sqrt{\sum_{t'=1}^{t} s_i^{t'}}} s_i^t$$

η is the learning rate which we set to 1. For relation identification training, we replace $\boldsymbol{\theta}$ and $\mathbf{f}(X, Y)$ in the above equations with $\boldsymbol{\psi}$ and

$$\mathbf{g}(X, Y) = \sum_{e \in E_G} \mathbf{g}(e).$$

We ran AdaGrad for ten iterations for concept identification, and five iterations for relation identification. The number of iterations was chosen by early stopping on the development set.

7 Experiments

We evaluate our parser on the newswire section of LDC2013E117 (deft-amr-release-r3-proxy.txt). Statistics about this corpus and our train/dev./test splits are given in Table 3.

[8]Because the alignments are automatic, some concepts may not be aligned, so we cannot compute their features. We remove the unaligned concepts and their edges from the full AMR graph for training. Thus some graphs used for training may in fact be disconnected.

Split	Document Years	Sentences	Tokens
Train	1995-2006	4.0k	79k
Dev.	2007	2.1k	40k
Test	2008	2.1k	42k

Table 3: Train/dev./test split.

Train			Test		
P	R	F_1	P	R	F_1
.92	.90	.91	.90	.79	.84

Table 4: Concept identification performance.

For the performance of concept identification, we report precision, recall, and F_1 of labeled spans using the induced labels on the training and test data as a gold standard (Table 4). Our concept identifier achieves 84% F_1 on the test data. Precision is roughly the same between train and test, but recall is worse on test, implicating unseen concepts as a significant source of errors on test data.

We evaluate the performance of the full parser using Smatch v1.0 (Cai and Knight, 2013), which counts the precision, recall and F_1 of the concepts and relations together. Using the full pipeline (concept identification and relation identification stages), our parser achieves 58% F_1 on the test data (Table 5). Using gold concepts with the relation identification stage yields a much higher Smatch score of 80% F_1. As a comparison, AMR Bank annotators have a consensus inter-annotator agreement Smatch score of 83% F_1. The runtime of our system is given in Figure 3.

The large drop in performance of 22% F_1 when moving from gold concepts to system concepts suggests that joint inference and training for the two stages might be helpful.

8 Related Work

Our approach to relation identification is inspired by graph-based techniques for non-projective syntactic dependency parsing. Minimum spanning tree algorithms—specifically, the optimum branching algorithm of Chu and Liu (1965) and Edmonds (1967)—were first used for dependency parsing by McDonald et al. (2005). Later ex-

	Train			Test		
concepts	P	R	F_1	P	R	F_1
gold	.85	.95	.90	.76	.84	.80
automatic	.69	.78	.73	.52	.66	.58

Table 5: Parser performance.

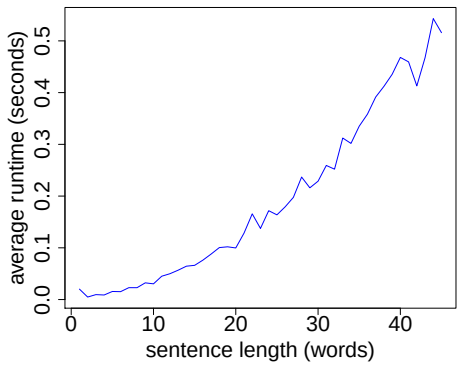


Figure 3: Runtime of JAMR (all stages).

tensions allow for *higher-order* (non–edge-local) features, often making use of relaxations to solve the NP-hard optimization problem. Mcdonald and Pereira (2006) incorporated second-order features, but resorted to an approximate algorithm. Others have formulated the problem as an integer linear program (Riedel and Clarke, 2006; Martins et al., 2009). TurboParser (Martins et al., 2013) uses AD^3 (Martins et al., 2011), a type of augmented Lagrangian relaxation, to integrate third-order features into a CLE backbone. Future work might extend JAMR to incorporate additional linguistically motivated constraints and higher-order features.

The task of concept identification is similar in form to the problem of Chinese word segmentation, for which semi-Markov models have successfully been used to incorporate features based on entire spans (Andrew, 2006).

While all semantic parsers aim to transform natural language text to a formal representation of its meaning, there is wide variation in the meaning representations and parsing techniques used. Space does not permit a complete survey, but we note some connections on both fronts.

Interlinguas (Carbonell et al., 1992) are an important precursor to AMR. Both formalisms are intended for use in machine translation, but AMR has an admitted bias toward the English language.

First-order logic representations (and extensions using, e.g., the λ-calculus) allow variable quantification, and are therefore more powerful. In recent research, they are often associated with combinatory categorial grammar (Steedman, 1996). There has been much work on statistical models for CCG parsing (Zettlemoyer and Collins, 2005; Zettlemoyer and Collins, 2007; Kwiatkowski et al., 2010, *inter alia*), usually using chart-based dynamic programming for inference.

Natural language interfaces for querying databases have served as another driving application (Zelle and Mooney, 1996; Kate et al., 2005; Liang et al., 2011, *inter alia*). The formalisms used here are richer in logical expressiveness than AMR, but typically use a smaller set of concept types—only those found in the database.

In contrast, semantic dependency parsing—in which the vertices in the graph correspond to the words in the sentence—is meant to make semantic parsing feasible for broader textual domains. Alshawi et al. (2011), for example, use shift-reduce parsing to map sentences to natural logical form.

AMR parsing also shares much in common with tasks like semantic role labeling and frame-semantic parsing (Gildea and Jurafsky, 2002; Punyakanok et al., 2008; Das et al., 2014, *inter alia*). In these tasks, predicates are often disambiguated to a canonical word sense, and roles are filled by spans (usually syntactic constituents). They consider each predicate separately, and produce a disconnected set of shallow predicate-argument structures. AMR, on the other hand, canonicalizes both predicates and arguments to a common concept label space. JAMR reasons about all concepts jointly to produce a unified representation of the meaning of an entire sentence.

9 Conclusion

We have presented the first published system for automatic AMR parsing, and shown that it provides a strong baseline based on the Smatch evaluation metric. We also present an algorithm for finding the maximum, spanning, connected subgraph and show how to incorporate extra constraints with Lagrangian relaxation. Our feature-based learning setup allows the system to be easily extended by incorporating new feature sources.

Acknowledgments

The authors gratefully acknowledge helpful correspondence from Kevin Knight, Ulf Hermjakob, and André Martins, and helpful feedback from Nathan Schneider, Brendan O'Connor, Waleed Ammar, and the anonymous reviewers. This work was sponsored by the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant number W911NF-10-1-0533 and DARPA grant FA8750-12-2-0342 funded under the DEFT program.

References

Hiyan Alshawi, Pi-Chuan Chang, and Michael Ringgaard. 2011. Deterministic statistical mapping of sentences to underspecified semantics. In *Proc. of ICWS*.

Galen Andrew. 2006. A hybrid markov/semi-markov conditional random field for sequence segmentation. In *Proc. of EMNLP*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proc. of the Linguistic Annotation Workshop and Interoperability with Discourse*.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proc. of ACL*.

Jaime G. Carbonell, Teruko Mitamura, and Eric H. Nyberg. 1992. The KANT perspective: A critique of pure transfer (and pure interlingua, pure transfer, ...). In *Proc. of the Fourth International Conference on Theoretical and Methodological Issues in Machine Translation: Empiricist vs. Rationalist Methods in MT*.

David Chiang, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, Bevan Jones, and Kevin Knight. 2013. Parsing graphs with hyperedge replacement grammars. In *Proc. of ACL*.

Y. J. Chu and T. H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proc. of EMNLP*.

Dipanjan Das, André F. T. Martins, and Noah A. Smith. 2012. An exact dual decomposition algorithm for shallow semantic parsing with constraints. In *Proc. of the Joint Conference on Lexical and Computational Semantics*.

Dipanjan Das, Desai Chen, André F. T. Martins, Nathan Schneider, and Noah A. Smith. 2014. Frame-semantic parsing. *Computational Linguistics*, 40(1):9–56.

Donald Davidson. 1967. The logical form of action sentences. In Nicholas Rescher, editor, *The Logic of Decision and Action*, pages 81–120. Univ. of Pittsburgh Press.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proc. of LREC*.

Bonnie Dorr, Nizar Habash, and David Traum. 1998. A thematic hierarchy for efficient generation from lexical-conceptual structure. In David Farwell, Laurie Gerber, and Eduard Hovy, editors, *Machine Translation and the Information Soup: Proc. of AMTA*.

Frank Drewes, Hans-Jörg Kreowski, and Annegret Habel. 1997. Hyperedge replacement graph grammars. In *Handbook of Graph Grammars*, pages 95–162. World Scientific.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, July.

Jack Edmonds. 1967. *Optimum branchings*. National Bureau of Standards.

Marshall L. Fisher. 2004. The Lagrangian relaxation method for solving integer programming problems. *Management Science*, 50(12):1861–1871.

Arthur M Geoffrion. 1974. *Lagrangean relaxation for integer programming*. Springer.

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

Jacques Janssen and Nikolaos Limnios. 1999. *Semi-Markov Models and Applications*. Springer, October.

Bevan Jones, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, and Kevin Knight. 2012. Semantics-based machine translation with hyperedge replacement grammars. In *Proc. of COLING*.

Rohit J. Kate, Yuk Wah Wong, and Raymond J. Mooney. 2005. Learning to transform natural to formal languages. In *Proc. of AAAI*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proc. of ACL*.

Joseph B. Kruskal. 1956. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proc. of the American Mathematical Society*, 7(1):48.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proc. of EMNLP*.

Percy Liang, Michael I. Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proc. of ACL*.

André F. T. Martins, Noah A. Smith, and Eric P. Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proc. of ACL*.

André F. T. Martins, Noah A. Smith, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. 2011. Dual decomposition with many overlapping components. In *Proc. of EMNLP*.

André F. T. Martins, Miguel Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective Turbo parsers. In *Proc. of ACL*.

Ryan Mcdonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. of EACL*, page 81–88.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of EMNLP*.

Terence Parsons. 1990. *Events in the Semantics of English: A study in subatomic semantics*. MIT Press.

Robert C. Prim. 1957. Shortest connection networks and some generalizations. *Bell System Technology Journal*, 36:1389–1401.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proc. of CoNLL*.

Sebastian Riedel and James Clarke. 2006. Incremental integer linear programming for non-projective dependency parsing. In *Proc. of EMNLP*.

Frank Rosenblatt. 1957. The perceptron–a perceiving and recognizing automaton. Technical Report 85-460-1, Cornell Aeronautical Laboratory.

Alexander M. Rush and Michael Collins. 2012. A tutorial on dual decomposition and Lagrangian relaxation for inference in natural language processing. *Journal of Artificial Intelligence Research*, 45(1):305—-362.

Mark Steedman. 1996. *Surface structure and interpretation*. Linguistic inquiry monographs. MIT Press.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proc. of AAAI*.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proc. of UAI*.

Luke S. Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *In Proc. of EMNLP-CoNLL*.

Context-dependent Semantic Parsing for Time Expressions

Kenton Lee[†], Yoav Artzi[†], Jesse Dodge[‡*], and Luke Zettlemoyer[†]

[†] Computer Science & Engineering, University of Washington, Seattle, WA
{kentonl, yoav, lsz}@cs.washington.edu

[‡] Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA
jessed@cs.cmu.edu

Abstract

We present an approach for learning context-dependent semantic parsers to identify and interpret time expressions. We use a Combinatory Categorial Grammar to construct compositional meaning representations, while considering contextual cues, such as the document creation time and the tense of the governing verb, to compute the final time values. Experiments on benchmark datasets show that our approach outperforms previous state-of-the-art systems, with error reductions of 13% to 21% in end-to-end performance.

1 Introduction

Time expressions present a number of challenges for language understanding systems. They have rich, compositional structure (e.g., "2nd Friday of July"), can be easily confused with non-temporal phrases (e.g., the word "May" can be a month name or a verb), and can vary in meaning in different linguistic contexts (e.g., the word "Friday" refers to different dates in the sentences "We met on *Friday*" and "We will meet on *Friday*"). Recovering the meaning of time expressions is therefore challenging, but provides opportunities to study context-dependent language use. In this paper, we present the first context-dependent semantic parsing approach for learning to identify and interpret time expressions, addressing all three challenges.

Existing state-of-the-art methods use hand-engineered rules for reasoning about time expressions (Strötgen and Gertz, 2013). This includes both *detection*, identifying a phrase as a time expression, and *resolution*, mapping such a phrase into a standardized time value. While rule-based approaches provide a natural way to express expert knowledge, it is relatively difficult to encode preferences between similar competing hypotheses and provide prediction confidence. Recently, methods for learning probabilistic semantic parsers have been shown to address such limitations (Angeli et al., 2012; Angeli and Uszkoreit, 2013). However, these approaches do not account for any surrounding linguistic context and were mainly evaluated with gold standard mentions.

We propose to use a context-dependent semantic parser for both detection and resolution of time expressions. For both tasks, we make use of a hand-engineered Combinatory Categorial Grammar (CCG) to construct a set of meaning representations that identify the time being described. For example, this grammar maps the phrase "2nd Friday of July" to the meaning representation $intersect(nth(2, friday), july)$, which encodes the set of all such days. Detection is then performed with a binary classifier to prune the set of text spans that can be parsed with the grammar (e.g., to tell that "born in *2000*" has a time expression but "a 2000 piece puzzle" does not). For resolution, we consider mentions sequentially and use a log-linear model to select the most likely meaning for each. This choice depends on contextual cues such as previous time expressions and the tense of the governing verb (e.g., as required to correctly resolve cases like "We should meet on the *2nd Friday of July*").

Such an approach provides a good balance between hand engineering and learning. For the relatively closed-class time expressions, we demonstrate that it is possible to engineer a high quality CCG lexicon. We take a data-driven approach for grammar design, preferring a grammar with high coverage even if it results in parsing ambiguities. We then learn a model to accurately select between competing parses and incorporate signals from the surrounding context, both more difficult to model with deterministic rules.

For both problems, we learn from TimeML an-

[*] Work conducted at the University of Washington.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1437–1447,
Baltimore, Maryland, USA, June 23-25 2014.

notations (Pustejovsky et al., 2005), which mark mentions and the specific times they reference. Training the detector is a supervised learning problem, but resolution is more challenging, requiring us to reason about latent parsing and context-dependent decisions.

We evaluate performance in two domains: the TempEval-3 corpus of newswire text (Uzzaman et al., 2013) and the WikiWars corpus of Wikipedia history articles (Mazur and Dale, 2010). On these benchmark datasets, we present new state-of-the-art results, with error reductions of up to 28% for the detection task and 21% for the end-to-end task.

2 Formal Overview

Time Expressions We follow the TIMEX3 standard (Pustejovsky et al., 2005) for defining time expressions within documents. Let a document $D = \langle w_1, \ldots, w_n \rangle$ be a sequence of n words w_i and a mention $m = (i, j)$ indicate start and end indices for a phrase $\langle w_i, \ldots, w_j \rangle$ in D. Define a time expression $e = (t, v)$ to include both a temporal type t and value v.[1] The temporal type $t \in \{\text{Date, Time, Duration, Set}\}$ can take one of four possible values, indicating if the expression e is a date (e.g., "January 10, 2014"), time (e.g., "11:59 pm"), duration (e.g., "6 months"), or set (e.g., "every year"). The value v is an extension of the ISO 8601 standard, which encodes the time that mention m refers to in the context provided by document D. For example, in a document written on Tuesday, January 7, 2014, "Friday," "three days later," and "January 10th" would all resolve to the value 2014-01-10. The time values are similarly defined for a wide range of expressions, such as underspecified dates (e.g., XXXX-01-10 for "Janunary 10th" when the year is not inferable from context) and durations (P2D for "two days").

Tasks Our goal is to find all time expressions in an input document. We divide the problem into two parts: detection and resolution. The *detection* problem is to take an input document D and output a mention set $M = \{m_i \mid i = 1 \ldots n\}$ of phrases in D that describe time expressions. The *resolution* problem (often also called *normalization*) is, given a document D and a set of mentions M, to map each $m \in M$ to the referred time expression e. This paper addresses both of these tasks.

Approach We learn separate, but related, models for detection and resolution. For both tasks, we define the space of possible compositional meaning representations $\mathcal{Z}$, where each $z \in \mathcal{Z}$ defines a unique time expression e. We use a log-linear CCG (Steedman, 1996; Clark and Curran, 2007) to rank possible meanings $z \in \mathcal{Z}$ for each mention m in a document D, as described in Section 4. Both detection (Section 5) and resolution (Section 6) rely on the semantic parser to identify likely mentions and resolve them within context. For learning we assume access to TimeML data containing documents labeled with time expressions. Each document D has a set $\{(m_i, e_i) | i = 1 \ldots n\}$, where each mention m_i marks a phrase that resolves to the time expression e_i.

Evaluation We evaluate performance (Section 8) for both newswire text and Wikipedia articles. We compare to the state-of-the-art systems for end-to-end resolution (Strötgen and Gertz, 2013) and resolution given gold mentions (Bethard, 2013b), both of which do not use any machine learning techniques.

3 Representing Time

We use simply typed lambda calculus to represent time expressions. Our representation draws heavily from the representation proposed by Angeli et al. (2012), who introduced semantic parsing for this task. There are five primitive types: duration d, sequence s, range r, approximate reference a, and numeral n, as described below. Table 1 lists the available constants for each type.

Duration A period of time. Each duration is a multiple of one of a closed set of possible base durations (e.g., $hour$, day, and $quarter$), which we refer to as its granularity. Table 1 includes the complete set of base durations used.

Range A specific interval of time, following an interval-based theory of time (Allen, 1981). The interval length is one of the base durations, which is the granularity of the range. Given two ranges R and R', we say that $R \subseteq R'$ if the endpoints of R lie on or within R'.

Sequence A set of ranges with identical granularity. The granularity of the sequence is that of its members. For example, $thursday$, which has a

[1] Time expressions also have optional modifier values for non-TIMEX properties (e.g., the modifier would contain $EARLY$ for the phrase "*early march*"). We do recover these modifiers but omit them from the discussion since they are not part of the official evaluation metrics.

Type	Primitive Constants
Duration	*second*, *minute*, *hour*, *timeofday*, *day*, *month*, *season*, *quarter*, *weekend*, *week*, *year*, *decade*, *century* , *temp_d*
Sequence	*monday*, *tuesday*, *wednesday*, *thursday*, *friday*, *saturday*, *sunday*, *january*, *february*, *march*, *april*, *may*, *june*, *july*, *august*, *september*, *october*, *november*, *december*, *winter*, *spring*, *summer*, *fall*, *night*, *morning*, *afternoon*, *evening*
Range	*ref_time*
Approximate reference	*present*, *future*, *past*, *unknown*
Numeral	*1*, *2*, *3*, *1999*, *2000*, *2001*, ...

Table 1: The types and primitive logical constants supported by the logical language for time.

day granularity, denotes the set of all day-granular ranges enclosing specific Thursdays. Given a range R and sequence S, we say that $R \in S$ if R is a member of S. Given two sequences S and S' we say that $S \subseteq S'$ if $R \in S$ implies $R \in S'$.

Approximate Reference An approximate time relative to the reference time. For example, $past$ and $future$. To handle mentions such as "a while," we add the constant $unknown$.

Numeral An integer, for example, 5 or 1990. Numerals are used to denote specific ranges, such as the year 2001, or to modify a duration's length.

Functions We also allow for functional types, for example $\langle s, r \rangle$ is assigned to a function that maps from sequences to ranges. Table 2 lists all supported functions with example mentions.

Context Dependent Constants To mark places where context-dependent choices will need to be made during resolution, we use two placeholder constants. First, ref_time denotes the mention reference time, which is later set to either the document time or a previously resolved mention. Second, $temp\_d$ is used in the $shift$ function to determine its return granularity, as described in Table 2, and is later replaced with the granularity of either the first or second argument of the enclosing $shift$ function. Section 4.3 describes how these decisions are made.

4 Parsing Time Expressions

We define a three-step derivation to resolve mentions to their TIMEX3 value. First, we use a CCG to generate an initial logical form for the mention. Next, we apply a set of operations that modify the initial logical form, as appropriate for its context. Finally, the logical form is resolved to a TIMEX3 value using a deterministic process.

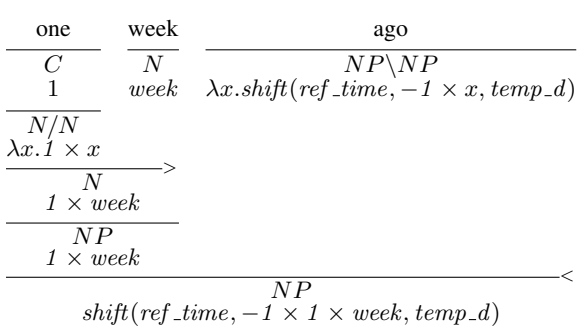


Figure 1: A CCG parse tree for the mention "one week ago." The tree includes forward (>) and backward (<) application, as well as two type-shifting operations

4.1 Combinatory Categorial Grammars

CCG is a linguistically motivated categorial formalism for modeling a wide range of language phenomena (Steedman, 1996; Steedman, 2000). A CCG is defined by a lexicon and a set of combinators. The lexicon pairs words with categories and the combinators define how to combine categories to create complete parse trees.

For example, Figure 1 shows a CCG parse tree for the phrase "one week ago." The parse tree is read top to bottom, starting from assigning categories to words using the lexicon. The lexical entry ago $\vdash NP\backslash NP : \lambda x.shift(ref\_time, -1 \times x, temp\_d)$ for the word "ago" pairs it with a category that has syntactic type $NP\backslash NP$ and semantics $\lambda x.shift(ref\_time, -1 \times x, temp\_d)$. Each intermediate parse node is then constructed by applying one of a small set of binary or unary operations (Steedman, 1996; Steedman, 2000), which modify both the syntax and semantics. We use backward (<) and forward (>) application and several unary type-shifting rules to handle number combinations. For example, in Figure 1 the category of the span "one week" is combined with the category of "ago" using backward application (<). Parsing concludes with a logical form representing the meaning of the complete mention.

Hand Engineered Lexicon To parse time expressions, we use a CCG lexicon that includes 287 manually designed entries, along with automatically generated entries such as numbers and common formats of dates and times. Figure 2 shows example entries from our lexicon.

Function	Description	Example
Operations on durations.		
$\times_{\langle n,\langle d,d\rangle\rangle}$	Given a duration D and a numeral N, return a duration D' that is N times longer than D.	"after *three days* of questioning" $3 \times day$
$some_{\langle d,d\rangle}$	Given a duration D, returns D', s.t. D' is the result of $D \times n$ for some $n > 1$.	"he left for *a few days*" $some(day)$
$seq_{\langle d,s\rangle}$	Given a duration D, generate a sequence S, s.t. S includes all ranges of type D.	"went to *last year*'s event" $previous(seq(year), ref\_time)$
Operations for extracting a specific range from a sequence.		
$this_{\langle s,\langle r,r\rangle\rangle}$	Given a sequence S and a range R, returns the range $R' \in S$, s.t. there exists a range R'' where $R \subseteq R''$ and $R' \subseteq R''$, and the length of R'' is minimal.	"a meeting *this friday*" $this(friday, ref\_time)$
$next_{\langle s,\langle r,r\rangle\rangle}$ $previous_{\langle s,\langle r,r\rangle\rangle}$	Given a sequence S and a range R, returns the range $R' \in S$ that is the one after/before $this(S, R)$.	"arriving *next month*" $next(seq(month), ref\_time)$
$nearest\_forward_{\langle s,\langle r,r\rangle\rangle}$ $nearest\_backward_{\langle s,\langle r,r\rangle\rangle}$	Given a sequence S and a range R, returns the range $R' \in S$ that is closest to R in the forward/backward direction.	"during *the coming weekend*" $nearest\_forward(seq(weekend), ref\_time)$
Operations for sequences.		
$nth_{\langle n,\langle s,\langle s,s\rangle\rangle\rangle}$ $nth_{\langle n,\langle s,s\rangle\rangle}$	Given a number N, a sequence S and a sequence S', returns a sequence $S'' \subseteq S$ s.t. for each $Q \in S''$ there exists $P \in S'$ and Q is the N-th entry in S that is a sub-interval of P. For the two-argument version, we use heuristics to infer the third argument by determining a sequence of higher granularity that is likely to contain the second argument.	"until *the second quarter of the year*" $nth(2, seq(quarter), seq(year))$
$intersect_{\langle s,\langle s,s\rangle\rangle}$	Given sequences S, S', where the duration of entries in S is shorter than these in S', return a sequence $S'' \subseteq S$, where for each $R \in S''$ there exists $R' \in S'$ s.t. $R \subseteq R'$.	"starts on *June 28*" $intersect(june, nth(28, seq(day), seq(month)))$
$shift_{\langle r,\langle d,\langle d,r\rangle\rangle\rangle}$	Given a range R, a duration D, and a duration G, return the range R', s.t. the starting point of R' is moved by the length of D. R' is converted to represent a range of granularity G by expanding if G has larger granularity, and is undefined if G has smaller granularity.	"*a week ago*, we went home" $shift(ref\_time, -1 \times 1 \times week, temp\_d)$
Operations on numbers.		
$\times_{\langle n,\langle n,n\rangle\rangle}$	Given two numerals, N' and N'', returns a numeral N''' representing their product $N' \times N''$.	"the battle lasted for *one hundred days*" $1 \times 100 \times day$
$+_{\langle n,\langle n,n\rangle\rangle}$	Given two numerals, N' and N'', returns a numeral N''' representing their sum $N' + N''$.	"open *twenty four hours*" $(20 + 4) \times hour$
Operations to mark sequences for specific TIMEX3 type annotations.		
$every_{\langle s,s\rangle}$	Given a sequence S, returns a sequence with SET temporal type.	"one dose *each day*" $every(seq(day))$
$bc_{\langle s,s\rangle}$	Convert a year to BC.	"during *five hundred BC*" $bc(nth(500, seq(year)))$

Table 2: Functional constants used to build logical expressions for representing time.

Manually Designed Entries:

several $\vdash NP/N : \lambda x.some(x)$
this $\vdash NP/N : \lambda x.this(x, ref\_time)$
each $\vdash NP/N : \lambda x.every(x)$
before $\vdash N\backslash NP/NP :$
$\lambda x.\lambda y.shift(x, -1 \times y, temp\_d)$
year $\vdash N : year$
wednesday $\vdash N : wednesday$
'20s $\vdash N : nth(192, seq(decade))$
yesterday $\vdash N : shift(ref\_time, -1 \times day, temp\_d)$

Automatically Generated Entries:

1992 $\vdash N : nth(1992, seq(year))$
nineteen ninety two $\vdash N : nth(1992, seq(year))$
09:30 $\vdash N : intersect(nth(10, seq(hour), seq(day)),$
$nth(31, seq(minute), seq(hour)))$
3rd $\vdash N\backslash N :$
$\lambda x.intersect(x, nth(3, seq(day), seq(month)))$

Figure 2: Example lexical entries.

4.2 Context-dependent Operations

To correctly resolve mentions to TIMEX3 values, the system must account for contextual information from various sources, including previous mentions in the document, the document creation time, and the sentence containing the mention. We consider three types of context operations, each takes as input a logical form z', modifies it and returns a new logical form z. Each context-dependent parse y specifies one operator of each type, which are applied to the logical form constructed by the CCG grammar, to produce the final, context-dependent logical form LF(y).

Reference Time Resolution The logical constant ref_time is replaced by either dct, representing the document creation time, or $last\_range$, the last r-typed mention resolved in the document. For example, consider the mention "the following year", which is represented using the logical form $next(seq(year), ref\_time)$. Within the sentence "*1998* was colder than *the following year*", the resolution of "the following year" depends on the previous mention "1998". In contrast, in "*The following year* will be warmer", its resolution depends on the document creation time.

Directionality Resolution If z' is s-typed we modify it to $nearest\_forward(z', ref\_time)$, $nearest\_backward(z', ref\_time)$, or z'. For example, given the sentence "...will be launched in *april*", the mention "april", and its logical form $april$, we would like to resolve it to the coming April, and therefore modify it to $nearest\_forward(april, ref\_time)$.

Shifting Granularity Every occurrence of the logical constant $temp\_d$, which is used as an argument to the function *shift* (see Table 2), is replaced with the granularity of either the first argument, the origin of the shift, or the second argument, the delta of the shift. This determines the final granularity of the output. For example, if the reference time is 2002-01, the mention "two years earlier" would resolve to either a month (since the reference time is of $month$ granularity) or a year (since the delta is of $year$ granularity).

4.3 Resolving Logical Forms

For a context-dependent parse y, we compute the TIMEX3 value TM(y) from the logical form $z =$ LF(y) with a deterministic step that performs a single traversal of z. Each primitive logical constant from Table 1 contributes to setting part of the TIMEX3 value (for example, specifying the day of the week) and the functional constants in Table 2 dictate transformations on the TIMEX3 values (for example, shifting forward or backward in time).[2]

5 Detection

The detection problem is to take an input document D and output a mention set $M = \{m_i \mid i = 1, \ldots, n\}$, where each mention m_i indexes a specific phrase in D that delimits a time expression.

Algorithm The detection algorithm considers all phrases that our CCG grammar Λ (Section 4) can parse, uses a learned classifier to further filter this set, and finally resolves conflicts between any overlapping predictions. We use a CKY algorithm to efficiently determine which phrases the CCG grammar can parse and only allow logical forms for which there exists some context in which they would produce a valid time expression, e.g. ruling out $intersect(monday, tuesday)$. Finally, we build the set M of non-overlapping mentions using a step similar to non-maximum suppression: the mentions are sorted by length (longest first) and iteratively added to M, as long as they do not overlap with any mention already in M.

Filtering Model Given a mention m, its document D, a feature function ϕ, the CCG lexicon Λ, and feature weights θ, we use a logistic regression model to define the probability distribution:

$$P(t|m, D; \Lambda, \theta) = \frac{e^{\theta \cdot \phi(m,D,\Lambda)}}{1 + e^{\theta \cdot \phi(m,D,\Lambda)}}$$

where t indicates whether m is a time expression.

Features We use three types of indicator features that test properties of the words in and around the potential mention m.

Context tokens Indicate the presence of a set of manually specified tokens near the mention. These include quotations around the mention, the word "old" after the mention, and prepositions of time (such as "in", "until", and "during") before.

Part of speech Indicators that pair each word with its part of speech, as assigned by the Stanford tagger (Toutanova et al., 2003).

Lexical group Each lexical entry belongs to one of thirteen manually defined lexical groups which cluster entries that contribute to the final time expression similarly. These groups include numbers, days of the week, months, seasons, etc. For each group, we include a feature indicating whether the parse includes a lexical entry from that group.

Determiner dependency Indicates the presence of a determiner in the mention and whether its parent in the dependency tree (generated by the Stanford parser (de Marneffe et al., 2006)) also resides within the mention.

Learning Finally, we construct the training data by considering all spans that (1) the CCG temporal grammar can parse and (2) are not strict subspans of an annotated mention. All spans that exactly matched the gold labels are used as positive examples and all others are negatives. Given this relaxed data, we learn the feature weights θ with L1-regularization. We set the probability threshold for detecting a time expression by optimizing the F1 score over the training data.

6 Resolution

The resolution problem is to, given a document D and a set of mentions M, map each $m \in M$ to the correct time expression e. Section 4 defined

[2] The full details are beyond the scope of this paper, but an implementation is available on the author's website.

the space of possible time expression that can be constructed for an input mention m in the context of a document D. In general, there will be many different possible derivations, and we will learn a model for selecting the best one.

Model Let y be a context-dependent CCG parse, which includes a parse tree TR(y), a set of context operations CNTX(y) applied to the logical form at the root of the tree, a final context-dependent logical form LF(y) and a TIMEX3 value TM(y). Define $\phi(m, D, y) \in \mathbb{R}^d$ to be a d-dimensional *feature–vector* representation and $\theta \in \mathbb{R}^d$ to be a parameter vector. The probability of a parse y for mention m and document D is:

$$P(y|m, D; \theta, \Lambda) = \frac{e^{\theta \cdot \phi(m,D,y)}}{\sum_{y'} e^{\theta \cdot \phi(m,D,y')}}$$

The inference problem at test time requires finding the best resolution by solving $y^*(m, D) = \arg\max_y P(y|m, D; \theta, \Lambda)$, where the final output TIMEX3 value is TM($y^*(m, D)$).

Inference We find the best context-dependent parse y by enumeration, as follows. We first parse the input mention m with a CKY-style algorithm, following previous work (Zettlemoyer and Collins, 2005). Due to the short length of time expressions and the manually constructed lexicon, we can perform exact inference. Given a parse, we then enumerate all possible outcomes for the context resolution operators. In practice, there are never more than one hundred possibilities.

Features The resolution features test properties of the linguistic context surrounding the mention m, relative to the context-dependent CCG parse y.

Governor verb We define the governor verb to be the nearest ancestor verb in the dependency parse of any token in m. We include features indicating the concatenation of the part-of-speech of the governor verb, its auxiliary verb if present, and the selected direction resolution operator (see Section 4.2). This feature helps to distinguish "They met on *Friday*" from "They will meet on *Friday*."

Temporal offset If the final logical form $LF(y)$ is a range, we define t to be the time difference between TM(y) and the reference time. For example, if the reference time is 2000-01-10 and the mention resolves to 2000-01-01, then t is -9 days. This feature indicates one of eleven bucketed values for t, including same day, less than a week, less than a month, etc. It allows the model to encode the likely temporal progression of a narrative. This feature is ignored if the granularity of TM(y) or the reference time is greater than a year.

Shift granularity The logical constant *shift* (Table 2) takes three arguments: the origin (range), the delta (duration), and the output granularity (duration). This indicator feature is the concatenation of each argument's granularity for every *shift* in $LF(y)$. It allows the model to determine whether "a year ago" refers to a year or a day.

Reference type Let r denote whether the reference time is the document creation time dct or the last range $last\_range$. Let g_l and g_r denote the granularities of $LF(y)$ and the reference time, respectively. We include features indicating the concatenations: $r{+}g_l$, $r{+}g_r$, and $r{+}g_l{+}g_r$. Additionally, we include features indicating the concatenation of r with each lexical entry used in the parse $TR(y)$. These features allow the model to encode preferences in selecting the correct reference time.

Fine-grained type These features indicate the fine-grained type of $TM(y)$, such as day of the month or week of the year. We also include a feature indicating the concatenation of each of these features with the direction resolution operator that was used. These features allow the model to represent, for example, that minutes of the year are less likely than days of the month.

Intersections These features indicate the concatenation of the granularities of any two sequences that appear as arguments to an $intersect$ constant.

Learning To estimate the model parameters θ we assume access to a set of training examples $\{(m_i, d_i, e_i) : i = 1, \ldots, n\}$, where each mention m_i is paired with a document d_i and a TIMEX3 value e_i. We use the AdaGrad algorithm (Duchi et al., 2011) to optimize the conditional, marginal log-likelihood of the data. For each mention, we marginalize over all possible context-dependent parses, using the predictions from the model on the previous gold mentions to fill in missing context, where necessary. After parameter estimation, we set a probability threshold for retaining a resolved time expression by optimizing value F1 (see Section 8) over the training data.

7 Related Work

Semantic parsers map sentences to logical representations of their underlying meaning, e.g., Zelle

and Mooney (1996), Zettlemoyer and Collins (2005), and Wong and Mooney (2007). Recently, research in this area has focused on learning for various forms of relatively weak but easily gathered supervision. This includes learning from question-answer pairs (Clarke et al., 2010; Liang et al., 2011; Kwiatkowski et al., 2013), from conversational logs (Artzi and Zettlemoyer, 2011), with distant supervision (Krishnamurthy and Mitchell, 2012; Cai and Yates, 2013), and from sentences paired with system behavior (Goldwasser and Roth, 2011; Chen and Mooney, 2011; Artzi and Zettlemoyer, 2013b). Recently, Angeli et al. introduced the idea of learning semantic parsers to resolve time expressions (Angeli et al., 2012) and showed that the approach can generalize to multiple languages (Angeli and Uszkoreit, 2013). Similarly, Bethard demonstrated that a hand-engineered semantic parser is also effective (Bethard, 2013b). However, these approaches did not use the semantic parser for detection and did not model linguistic context during resolution.

We build on a number of existing algorithmic ideas, including using CCGs to build meaning representations (Zettlemoyer and Collins, 2005; Zettlemoyer and Collins, 2007; Kwiatkowski et al., 2010; Kwiatkowski et al., 2011), building derivations to transform the output of the CCG parser based on context (Zettlemoyer and Collins, 2009), and using weakly supervised parameter updates (Artzi and Zettlemoyer, 2011; Artzi and Zettlemoyer, 2013b). However, we are the first to use a semantic parsing grammar within a mention detection algorithm, thereby avoiding the need to represent the meaning of complete sentences, and the first to develop a context-dependent model for semantic parsing of time expressions.

Time expressions have been extensively studied as part of the TimeEx task, including 9 teams who competed in the 2013 TempEval-3 competition (Uzzaman et al., 2013). This line of work builds on ideas from TimeBank (Pustejovsky et al., 2003) and a number of different formal models for temporal reasoning, e.g. Allen (1983), Moens and Steedman (1988). In 2013, HeidelTime (Strötgen and Gertz, 2013) was the top performing system. It used deterministic rules defined over regular expressions to perform both detection and resolution, and will provide a comparison system for our evaluation in Section 9. In general, many different rule-based systems, e.g. NavyTime (Chambers, 2013) and SUTime (Chang and Manning, 2012), and learning systems, e.g. ClearTK (Bethard, 2013a) and MANTime (Filannino et al., 2013), did well for detection. However, rule-based approaches dominated in resolution; none of the top performers attempted to learn to do resolution. Our approach is a hybrid of rule based and learning, by using latent-variable learning techniques to estimate CCG parsing and context resolution models from the provided data.

Corpus	Doc.	Token	TimeEx
TempEval-3 (Dev)	256	95,391	1,822
TempEval-3 (Test)	20	6,375	138
WikiWars (Dev)	17	98,746	2,228
WikiWars (Test)	5	19,052	363

Figure 3: Corpus statistics.

8 Experimental Setup

Data We evaluate performance on the TempEval-3 (Uzzaman et al., 2013) and WikiWars (Mazur and Dale, 2010) datasets. Figure 3 shows summary statistics for both datasets. For the TempEval-3 corpus, we use the given training and testing set splits. Since the training set has lower inter-annotator agreement than the testing set (Uzzaman et al., 2013), we manually corrected all of the mistakes we found in the training data.[3] The original training set is denoted Dev* and the corrected Dev. We report (1) cross-validation development results on Dev*, (2) cross-validation development and ablation results for Dev, and (3) held-out test results after training with Dev. For WikiWars, we randomly assigned the data to include 17 training documents (2,228 time expressions) and 5 test documents (363 time expressions). We use cross-validation on the training data for development. All cross-validation experiments used 10 folds.

Implementation Our system was implemented using the open source University of Washington Semantic Parsing Framework (Artzi and Zettlemoyer, 2013a). We used LIBLINEAR (Fan et al., 2008) to learn the detection model.

Parameter Settings We use the same set of parameters for both datasets, chosen based on development experiments. For detection, we set the regularization parameter to 10 with a stopping crite-

[3]We modified the annotations for 18% of the mentions. This relabeled corpus is available on the author's website.

	System	Strict Detection			Relaxed Detection			Type Res.		Value Resolution			
		Pre.	Rec.	**F1**	Pre.	Rec.	**F1**	Acc.	**F1**	Acc.	Pre.	Rec.	**F1**
Dev*	This work	84.6	83.4	**84.0**	92.8	91.5	**92.1**	94.6	**87.1**	84.0	77.9	76.8	**77.4**
	HeidelTime	83.7	83.4	83.5	91.7	91.4	91.6	95.0	87.0	84.1	77.1	76.8	77.0
Dev	This work	92.7	89.6	**91.1**	97.4	94.1	**95.7**	97.1	**92.9**	91.5	89.1	86.1	**87.6**
	Context ablation	92.7	89.3	91.0	97.5	93.9	95.7	97.1	92.9	89.8	87.6	84.3	85.9
	HeidelTime	90.2	84.8	87.4	96.5	90.7	93.5	96.1	89.9	88.4	85.3	80.2	82.7
Test	This work	86.1	80.4	**83.1**	94.6	88.4	**91.4**	93.4	**85.4**	90.2	85.3	79.7	**82.4**
	HeidelTime	83.9	79.0	81.3	93.1	87.7	90.3	90.9	82.1	86.0	80.1	75.4	77.7
	NavyTime	78.7	80.4	79.6	89.4	91.3	90.3	88.9	80.3	78.6	70.3	71.8	71.0
	ClearTK	85.9	79.7	82.7	93.8	87.0	90.2	93.3	84.2	71.7	67.3	62.4	64.7

Figure 4: TempEval-3 development and test results, compared to the top systems in the shared task.

	System	Strict Detection			Relaxed Detection			Value Resolution			
		Pre.	Rec.	**F1**	Pre.	Rec.	**F1**	Acc.	Pre.	Rec.	**F1**
Dev	This work	90.3	83.0	**86.5**	98.1	90.1	**93.9**	87.6	85.9	78.9	**82.3**
	Context ablation	90.9	80.1	85.2	98.2	86.5	92.0	68.5	67.3	59.3	63.0
	HeidelTime	86.0	75.3	80.3	95.4	83.5	89.0	90.5	86.3	75.6	80.6
Test	This work	87.7	78.8	**83.0**	97.6	87.6	**92.3**	84.6	82.5	74.1	**78.1**
	HeidelTime	85.2	79.3	82.1	92.6	86.2	89.3	83.7	77.5	72.1	74.7

Figure 5: WikiWars development and test results.

rion of 0.01. For resolution, we set the learning rate to 0.25 and ran AdaGrad for 5 iterations. All features are initialized to have zero weights.

Evaluation Metrics We use the official TempEval-3 scoring script and report the standard metrics. We report *detection precision, recall* and *F1* with relaxed and strict metrics; a gold mention is considered detected for the relaxed metric if any of the output candidates overlap with it and is detected for the strict metric if the extent of any output candidates matches exactly. For resolution, we report *value accuracy*, measuring correctness of time expressions detected according to the relaxed metric. We also report *value precision, recall,* and *F1*, which score an expression as correct if it is both correctly detected (relaxed) and resolved. For end-to-end performance, value F1 is the primary metric. Finally, we report accuracy and F1 for temporal types, as defined in Section 2, for the TempEval dataset (WikiWars does not include type labels).

Comparison Systems We compare our system primarily to HeidelTime (Strötgen and Gertz, 2013), which is state of the art in the end-to-end task. For the TempEval-3 dataset, we also compare to two other strong participants of the shared task. These include NavyTime (Chambers, 2013), which had the top relaxed detection score, and ClearTK (Bethard, 2013a), which had the top strict detection score and type F1 score. We also include a comparison with Bethard's synchronous context free grammar (SCFG) (Bethard, 2013b), which is state-of-the-art in the task of resolution with gold mention boundaries.

System	Dev*	Dev	Test
This work	**81.8**	**90.1**	**82.6**
SCFG	77.0	81.6	78.9

Figure 6: TempEval-3 gold mention value accuracy.

9 Results

End-to-end results Figure 4 shows development and test results for TempEval-3. Figure 5 shows these numbers for WikiWars. In both datasets, we achieve state-of-the-art test scores. For detection, we show up to 3-point improvements in strict and relaxed F1 scores. These numbers outperform all systems participating in the shared task, which used a variety of techniques including hand-engineered rules, CRF tagging models, and SVMs. For resolution, we show up to 4-point improvements in the value F1 score, also outperforming participating systems, all of which used hand-engineered rules for resolution.

Gold Mentions Figure 6 reports development and test results with gold mentions.[4] Our approach outperforms the state of the art, SCFG (Bethard, 2013b), which also used a hand engineered grammar, but did not use machine learning techniques.

[4]These numbers vary slightly from those reported; we did not count the document creation times as mentions.

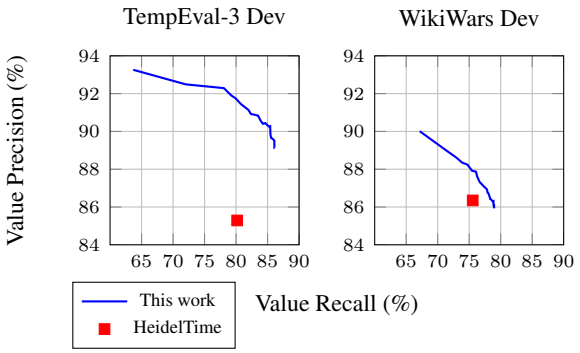


Figure 7: Value precision vs. recall for 10-fold cross validation on TempEval-3 Dev and WikiWars Dev.

Precision vs. Recall Our probabilistic model of time expression resolution allows us to easily tradeoff precision and recall for end-to-end performance by varying the resolution probability threshold. Figure 7 shows the precision vs. recall of the resolved values from 10-fold cross validation of TempEval-3 Dev and WikiWars Dev. We are able to achieve precision at or above 90% with reasonable recall, nearly 70% for WikiWars and over 85% for TempEval-3.

Ablation Study Figures 4-5 also show comparisons for our system with no context. We ablate the ability to refer to the context during resolution by removing contextual information from the resolution features and only allowing the document creation time to be the reference time.

We see an interesting asymmetry in the effect of modeling context across the two domains. We find that context is much more important in WikiWars (19 point difference) than in TempEval (2 point difference). This result reaffirms the difference in domains that Strötgen and Gertz (2012) noted during the development of HeidelTime: history articles have narrative structure that moves back and forth through time while newspaper text typically describes events happening near the document creation time. This difference helps us to understand why previous learning systems have been able to ignore context and perform well on newswire text.

Error Analysis To investigate the source of error, we compute oracle results for resolving gold mentions over the TempEval-3 Dev dataset. We found that our system produces a correct candidate derivation for 96% of the mentions.

We also manually categorized all resolution errors for end-to-end performance with 10-fold cross validation of the TempEval-3 Dev dataset, shown in Figure 8. The lexicon allows for effective parsing, contributing to only 2% of the overall errors. However, context is more challenging. The three largest categories, responsible for 64.7% of the errors, were incorrect use of the context operators. More expressive modeling will be required to fully capture the complex pragmatics involved in understanding time expressions.

Error description	%
Wrong directionality context operator	34.6
Wrong reference time context operator	15.7
Wrong shifting granularity context operator	14.4
Requires joint reasoning with events	9.2
Cascading error due to wrong detection	7.8
CCG parse error	2.0
Other error	16.3

Figure 8: Resolution errors from 10-fold cross validation of the TempEval-3 Dev dataset.

10 Conclusion

We presented the first context-dependent semantic parsing system to detect and resolve time expressions. Both models used a Combinatory Categorial Grammar (CCG) to construct a set of possible temporal meaning representations. This grammar defined the possible phrases for detection and the inputs to a context-dependent reasoning step that was used to construct the output time expression during resolution. Experiments demonstrated that our approach outperforms state-of-the-art systems.

In the future, we aim to develop joint models for reasoning about events and time expressions, including detection and resolution of temporal relations. We are also interested in testing coverage in new domains and investigating techniques for semi-supervised learning and learning with noisy data. We hypothesize that semantic parsing techniques could help in all of these settings, providing a unified mechanism for compositional analysis within temporal understanding problems.

Acknowledgments

The research was supported in part by DARPA under the DEFT program through the AFRL (FA8750-13-2-0019) and the CSSG (N11AP20020), and the NSF (IIS-1115966, IIS-1252835). The authors thank Nicholas FitzGerald, Tom Kwiatkowski, and Mark Yatskar for helpful discussions, and the anonymous reviewers for helpful comments.

References

James F. Allen. 1981. An interval-based representation of temporal knowledge. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*.

James F Allen. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.

Gabor Angeli and Jakob Uszkoreit. 2013. Language-independent discriminative parsing of temporal expressions. In *Proceedings of the Conference of the Association of Computational Linguistics*.

Gabor Angeli, Christopher D Manning, and Daniel Jurafsky. 2012. Parsing time: Learning to interpret time expressions. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*.

Y. Artzi and L.S. Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Y. Artzi and L.S. Zettlemoyer. 2013a. UW SPF: The University of Washington Semantic Parsing Framework.

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

Steven Bethard. 2013a. Cleartk-timeml: A minimalist approach to tempeval 2013. In *Second Joint Conference on Lexical and Computational Semantics*.

Steven Bethard. 2013b. A synchronous context free grammar for time normalization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Q. Cai and A. Yates. 2013. Semantic parsing freebase: Towards open-domain semantic parsing. In *Joint Conference on Lexical and Computational Semantics: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*.

Nathanael Chambers. 2013. Navytime: Event and time ordering from raw text. In *Second Joint Conference on Lexical and Computational Semantics*.

Angel X Chang and Christopher Manning. 2012. Sutime: A library for recognizing and normalizing time expressions. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*.

D.L. Chen and R.J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the National Conference on Artificial Intelligence*.

S. Clark and J. R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

J. Clarke, D. Goldwasser, M. Chang, and D. Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of the Conference on Computational Natural Language Learning*.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Michele Filannino, Gavin Brown, and Goran Nenadic. 2013. Mantime: Temporal expression identification and normalization in the tempeval-3 challenge. In *Second Joint Conference on Lexical and Computational Semantics*.

D. Goldwasser and D. Roth. 2011. Learning from natural instructions. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

J. Krishnamurthy and T. Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

T. Kwiatkowski, L.S. Zettlemoyer, S. Goldwater, and M. Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

T. Kwiatkowski, L.S. Zettlemoyer, S. Goldwater, and M. Steedman. 2011. Lexical Generalization in CCG Grammar Induction for Semantic Parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

T. Kwiatkowski, E. Choi, Y. Artzi, and L. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

P. Liang, M.I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the Conference of the Association for Computational Linguistics*.

Pawet Mazur and Robert Dale. 2010. Wikiwars: a new corpus for research on temporal expressions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Marc Moens and Mark Steedman. 1988. Temporal ontology and temporal reference. *Computational linguistics*, 14(2):15–28.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, et al. 2003. The timebank corpus. In *Corpus linguistics*.

James Pustejovsky, Bob Ingria, Roser Sauri, Jose Castano, Jessica Littman, Rob Gaizauskas, Andrea Setzer, Graham Katz, and Inderjeet Mani. 2005. The specification language timeml. *The language of time: A reader*, pages 545–557.

M. Steedman. 1996. *Surface Structure and Interpretation*. The MIT Press.

M. Steedman. 2000. *The Syntactic Process*. The MIT Press.

Jannik Strötgen and Michael Gertz. 2012. Temporal tagging on different domains: Challenges, strategies, and gold standards. In *Proceedings of the Eigth International Conference on Language Resources and Evaluation*.

Jannik Strötgen and Michael Gertz. 2013. Multilingual and cross-domain temporal tagging. *Language Resources and Evaluation*, 47(2):269–298.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*.

N. Uzzaman, H. Llorens, L. Derczynski, M. Verhagen, J. Allen, and J. Pustejovsky. 2013. Semeval-2013 task 1: Tempeval-3: Evaluating time expressions, events, and temporal relations. In *Proceedings of the International Workshop on Semantic Evaluation*.

Y.W. Wong and R.J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the Conference of the Association for Computational Linguistics*.

J.M. Zelle and R.J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the National Conference on Artificial Intelligence*.

L.S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

L.S. Zettlemoyer and M. Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

L.S. Zettlemoyer and M. Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proceedings of the Joint Conference of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing*.

Semantic Frame Identification with Distributed Word Representations

Karl Moritz Hermann[‡*] **Dipanjan Das**[†] **Jason Weston**[†] **Kuzman Ganchev**[†]
[‡]Department of Computer Science, University of Oxford, Oxford OX1 3QD, United Kingdom
[†] Google Inc., 76 9th Avenue, New York, NY 10011, United States
karl.moritz.hermann@cs.ox.ac.uk
{dipanjand,kuzman}@google.com jaseweston@gmail.com

Abstract

We present a novel technique for semantic frame identification using distributed representations of predicates and their syntactic context; this technique leverages automatic syntactic parses and a generic set of word embeddings. Given labeled data annotated with frame-semantic parses, we learn a model that projects the set of word representations for the syntactic context around a predicate to a low dimensional representation. The latter is used for semantic frame identification; with a standard argument identification method inspired by prior work, we achieve state-of-the-art results on FrameNet-style frame-semantic analysis. Additionally, we report strong results on PropBank-style semantic role labeling in comparison to prior work.

1 Introduction

Distributed representations of words have proved useful for a number of tasks. By providing richer representations of meaning than what can be encompassed in a discrete representation, such approaches have successfully been applied to tasks such as sentiment analysis (Socher et al., 2011), topic classification (Klementiev et al., 2012) or word-word similarity (Mitchell and Lapata, 2008).

We present a new technique for semantic **frame** identification that leverages distributed word representations. According to the theory of frame semantics (Fillmore, 1982), a semantic frame represents an event or scenario, and possesses frame elements (or semantic **roles**) that participate in the event. Most work on frame-semantic parsing has usually divided the task into two major subtasks: *frame identification*, namely the disambiguation of a given predicate to a frame, and *argument identification* (or semantic role labeling), the analysis of words and phrases in the sentential context that satisfy the frame's semantic roles (Das et al., 2010; Das et al., 2014).[1] Here, we focus on the first subtask of frame identification for given predicates; we use our novel method (§3) in conjunction with a standard argument identification model (§4) to perform full frame-semantic parsing.

We present experiments on two tasks. First, we show that for frame identification on the FrameNet corpus (Baker et al., 1998; Fillmore et al., 2003), we outperform the prior state of the art (Das et al., 2014). Moreover, for full frame-semantic parsing, with the presented frame identification technique followed by our argument identification method, we report the best results on this task to date. Second, we present results on PropBank-style semantic role labeling (Palmer et al., 2005; Meyers et al., 2004; Màrquez et al., 2008), that approach strong baselines, and are on par with prior state of the art (Punyakanok et al., 2008).

2 Overview

Early work in frame-semantic analysis was pioneered by Gildea and Jurafsky (2002). Subsequent work in this area focused on either the FrameNet or PropBank frameworks, and research on the latter has been more popular. Since the CoNLL 2004-2005 shared tasks (Carreras and Màrquez,

[*]The majority of this research was carried out during an internship at Google.

[1]There are exceptions, wherein the task has been modeled using a pipeline of three classifiers that perform frame identification, a binary stage that classifies candidate arguments, and argument identification on the filtered candidates (Baker et al., 2007; Johansson and Nugues, 2007).

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1448–1458,
Baltimore, Maryland, USA, June 23-25 2014.

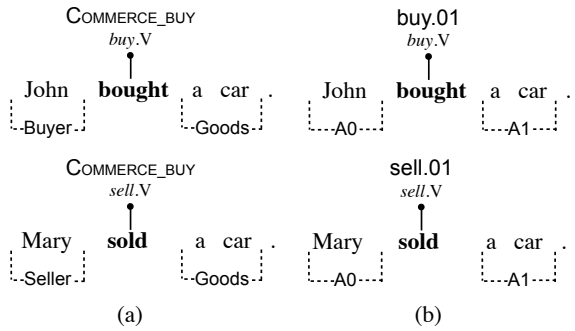


Figure 1: Example sentences with frame-semantic analyses. FrameNet annotation conventions are used in (a) while (b) denotes PropBank conventions.

2004; Carreras and Màrquez, 2005) on PropBank semantic role labeling (SRL), it has been treated as an important NLP problem. However, research has mostly focused on argument analysis, skipping the frame disambiguation step, and its interaction with argument identification.

2.1 Frame-Semantic Parsing

Closely related to SRL, frame-semantic parsing consists of the resolution of predicate sense into a frame, and the analysis of the frame's arguments. Work in this area exclusively uses the FrameNet full text annotations. Johansson and Nugues (2007) presented the best performing system at SemEval 2007 (Baker et al., 2007), and Das et al. (2010) improved performance, and later set the current state of the art on this task (Das et al., 2014). We briefly discuss FrameNet, and subsequently PropBank annotation conventions here.

FrameNet The FrameNet project (Baker et al., 1998) is a lexical database that contains information about words and phrases (represented as lemmas conjoined with a coarse part-of-speech tag) termed as lexical units, with a set of semantic frames that they could evoke. For each frame, there is a list of associated frame elements (or roles, henceforth), that are also distinguished as core or non-core.[2] Sentences are annotated using this universal frame inventory. For example, consider the pair of sentences in Figure 1(a). COMMERCE_BUY is a frame that can be evoked by morphological variants of the two example lexical units *buy*.V and *sell*.V. Buyer, Seller and Goods are some example roles for this frame.

[2]Additional information such as finer distinction of the coreness properties of roles, the relationship between frames, and that of roles are also present, but we do not leverage that information in this work.

PropBank The PropBank project (Palmer et al., 2005) is another popular resource related to semantic role labeling. The PropBank corpus has verbs annotated with sense frames and their arguments. Like FrameNet, it also has a lexical database that stores type information about verbs, in the form of sense frames and the possible semantic roles each frame could take. There are modifier roles that are shared across verb frames, somewhat similar to the non-core roles in FrameNet. Figure 1(b) shows annotations for two verbs "bought" and "sold", with their lemmas (akin to the lexical units in FrameNet) and their verb frames buy.01 and sell.01. Generic core role labels (of which there are seven, namely A0-A5 and AA) for the verb frames are marked in the figure.[3] A key difference between the two annotation systems is that PropBank uses a local frame inventory, where frames are predicate-specific. Moreover, role labels, although few in number, take specific meaning for each verb frame. Figure 1 highlights this difference: while both *sell*.V and *buy*.V are members of the same frame in FrameNet, they evoke different frames in PropBank. In spite of this difference, nearly identical statistical models could be employed for both frameworks.

Modeling In this paper, we model the frame-semantic parsing problem in two stages: **frame identification** and **argument identification**. As mentioned in §1, these correspond to a frame disambiguation stage,[4] and a stage that finds the various arguments that fulfill the frame's semantic roles within the sentence, respectively. This resembles the framework of Das et al. (2014), who solely focus on FrameNet corpora, unlike this paper. The novelty of this paper lies in the frame identification stage (§3). Note that this two-stage approach is unusual for the PropBank corpora when compared to prior work, where the vast majority of published papers have not focused on the verb frame disambiguation problem at all, only focusing on the role labeling stage (see the overview paper of Màrquez et al. (2008) for example).

[3]NomBank (Meyers et al., 2004) is a similar resource for nominal predicates, but we do not consider it in our experiments.

[4]For example in PropBank, the lexical unit *buy*.V has three verb frames and in sentential context, we want to disambiguate its frame. (Although PropBank never formally uses the term lexical unit, we adopt its usage from the frame semantics literature.)

2.2 Distributed Frame Identification

We present a model that takes word embeddings as input and learns to identify semantic frames. A word embedding is a distributed representation of meaning where each word is represented as a vector in $\mathbb{R}^n$. Such representations allow a model to share meaning between similar words, and have been used to capture semantic, syntactic and morphological content (Collobert and Weston, 2008; Turian et al., 2010, *inter alia*). We use word embeddings to represent the syntactic context of a particular predicate instance as a vector. For example, consider the sentence "He runs the company." The predicate *runs* has two syntactic dependents – a subject and direct object (but no prepositional phrases or clausal complements). We could represent the syntactic context of *runs* as a vector with blocks for all the possible dependents warranted by a syntactic parser; for example, we could assume that positions $0 \ldots n$ in the vector correspond to the subject dependent, $n{+}1 \ldots 2n$ correspond to the clausal complement dependent, and so forth. Thus, the context is a vector in $\mathbb{R}^{nk}$ with the embedding of *He* at the subject position, the embedding of *company* in direct object position and zeros everywhere else. Given input vectors of this form for our training data, we learn a matrix that maps this high dimensional and sparse representation into a lower dimensional space. Simultaneously, the model learns an embedding for all the possible labels (i.e. the frames in a given lexicon). At inference time, the predicate-context is mapped to the low dimensional space, and we choose the nearest frame label as our classification. We next describe this model in detail.

3 Frame Identification with Embeddings

We continue using the example sentence from §2.2: "He runs the company." where we want to disambiguate the frame of *runs* in context. First, we extract the words in the syntactic context of *runs*; next, we concatenate their word embeddings as described in §2.2 to create an initial vector space representation. Subsequently, we learn a mapping from this initial representation into a low-dimensional space; we also learn an embedding for each possible frame label in the same low-dimensional space. The goal of learning is to make sure that the correct frame label is as close as possible to the mapped context, while competing frame labels are farther away.

Formally, let x represent the actual sentence with a marked predicate, along with the associated syntactic parse tree; let our initial representation of the predicate context be $g(x)$. Suppose that the word embeddings we start with are of dimension n. Then g is a function from a parsed sentence x to $\mathbb{R}^{nk}$, where k is the number of possible syntactic context types. For example g selects some important positions relative to the predicate, and reserves a block in its output space for the embedding of words found at that position. Suppose g considers clausal complements and direct objects. Then $g : X \rightarrow \mathbb{R}^{2n}$ and for the example sentence it has zeros in positions $0 \ldots n$ and the embedding of the word *company* in positions $n{+}1 \ldots 2n$.

$$g(x) = [0, \ldots, 0, \text{embedding of } \textit{company}].$$

Section 3.1 describes the context positions we use in our experiments. Let the low dimensional space we map to be $\mathbb{R}^m$ and the learned mapping be $M : \mathbb{R}^{nk} \rightarrow \mathbb{R}^m$. The mapping M is a linear transformation, and we learn it using the WSABIE algorithm (Weston et al., 2011). WSABIE also learns an embedding for each frame label (y, henceforth). In our setting, this means that each frame corresponds to a point in $\mathbb{R}^m$. If we have F possible frames we can store those parameters in an $F \times m$ matrix, one m-dimensional point for each frame, which we will refer to as the linear mapping Y. Let the lexical unit (the lemma conjoined with a coarse POS tag) for the marked predicate be ℓ. We denote the frames that associate with ℓ in the frame lexicon[5] and our training corpus as F_ℓ. WSABIE performs gradient-based updates on an objective that tries to minimize the distance between $M(g(x))$ and the embedding of the correct label $Y(y)$, while maintaining a large distance between $M(g(x))$ and the other possible labels $Y(\bar{y})$ in the confusion set F_ℓ. At disambiguation time, we use a simple dot product similarity as our distance metric, meaning that the model chooses a label by computing the $\text{argmax}_y s(x, y)$ where $s(x, y) = M(g(x)) \cdot Y(y)$, where the argmax iterates over the possible frames $y \in F_\ell$ if ℓ was seen in the lexicon or the training data, or $y \in F$, if it was unseen.[6] Model learning is performed using the margin ranking loss function as described in

[5]The frame lexicon stores the frames, corresponding semantic roles and the lexical units associated with the frame.

[6]This disambiguation scheme is similar to the one adopted by Das et al. (2014), but they use unlemmatized words to define their confusion set.

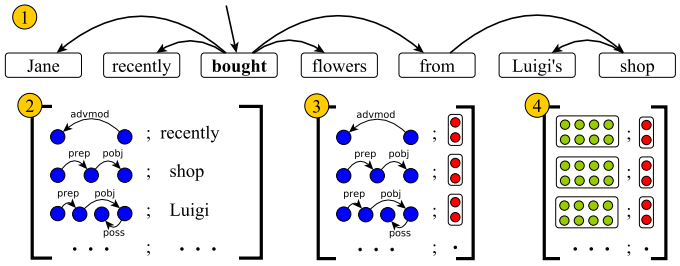


Figure 2: Context representation extraction for the embedding model. Given a dependency parse (1) the model extracts all words matching a set of paths from the frame evoking predicate and its direct dependents (2). The model computes a composed representation of the predicate instance by using distributed vector representations for words (3) – the (red) vertical embedding vectors for each word are concatenated into a long vector. Finally, we learn a linear transformation function parametrized by the context blocks (4).

Weston et al. (2011), and in more detail in section 3.2.

Since WSABIE learns a single mapping from $g(x)$ to $\mathbb{R}^m$, parameters are shared between different words and different frames. So for example "He *runs* the company" could help the model disambiguate "He *owns* the company." Moreover, since $g(x)$ relies on word embeddings rather than word identities, information is shared between words. For example "*He* runs *the company*" could help us to learn about "*She* runs *a corporation*".

3.1 Context Representation Extraction

In principle $g(x)$ could be any feature function, but we performed an initial investigation of two particular variants. In both variants, our representation is a block vector where each block corresponds to a syntactic position relative to the predicate, and each block's values correspond to the embedding of the word at that position.

Direct Dependents The first context function we considered corresponds to the examples in §3. To elaborate, the positions of interest are the labels of the direct dependents of the predicate, so k is the number of labels that the dependency parser can produce. For example, if the label on the edge between *runs* and *He* is nsubj, we would put the embedding of *He* in the block corresponding to nsubj. If a label occurs multiple times, then the embeddings of the words below this label are averaged.

Unfortunately, using only the direct dependents can miss a lot of useful information. For example, topicalization can place discriminating information farther from the predicate. Consider "He *runs* the *company*." vs. "It was the *company* that he *runs*." In the second sentence, the discriminating word, *company* dominates the predicate *runs*. Similarly, predicates in embedded clauses may have a distant agent which cannot be captured using direct dependents. Consider "*The athlete ran* the marathon." vs. "*The athlete* prepared himself for three months to *run* the marathon." In the second example, for the predicate *run*, the agent *The athlete* is not a direct dependent, but is connected via a longer dependency path.

Dependency Paths To capture more relevant context, we developed a second context function as follows. We scanned the training data for a given task (either the PropBank or the FrameNet domains) for the dependency paths that connected the gold predicates to the gold semantic arguments. This set of dependency paths were deemed as possible positions in the initial vector space representation. In addition, akin to the first context function, we also added all dependency labels to the context set. Thus for this context function, the block cardinality k was the sum of the number of scanned gold dependency path types and the number of dependency labels. Given a predicate in its sentential context, we therefore extract only those context words that appear in positions warranted by the above set. See Figure 2 for an illustration of this process.

We performed initial experiments using context extracted from 1) direct dependents, 2) dependency paths, and 3) both. For all our experiments, setting 3) which concatenates the direct dependents and dependency path always dominated the other two, so we only report results for this setting.

3.2 Learning

We model our objective function following Weston et al. (2011), using a weighted approximate-rank pairwise loss, learned with stochastic gradient descent. The mapping from $g(x)$ to the low dimensional space $\mathbb{R}^m$ is a linear transformation, so the model parameters to be learnt are the matrix $M \in \mathbb{R}^{nk \times m}$ as well as the embedding of each possible frame label, represented as another matrix $Y \in \mathbb{R}^{F \times m}$ where there are F frames in total. The training objective function minimizes:

$$\sum_x \sum_{\bar{y}} L\big(rank_y(x)\big) \max(0, \gamma + s(x,y) - s(x,\bar{y})).$$

where x, y are the training inputs and their corresponding correct frames, and $\bar{y}$ are negative frames, γ is the margin. Here, $rank_y(x)$ is the rank of the positive frame y relative to all the negative frames:

$$rank_y(x) = \sum_{\bar{y}} I(s(x,y) \leq \gamma + s(x,\bar{y})),$$

and $L(\eta)$ converts the rank to a weight. Choosing $L(\eta) = C\eta$ for any positive constant C optimizes the mean rank, whereas a weighting such as $L(\eta) = \sum_{i=1}^{\eta} 1/i$ (adopted here) optimizes the top of the ranked list, as described in (Usunier et al., 2009). To train with such an objective, stochastic gradient is employed. For speed the computation of $rank_y(x)$ is then replaced with a sampled approximation: sample N items $\bar{y}$ until a violation is found, i.e. $\max(0, \gamma + s(x,\bar{y}) - s(x,y))) > 0$ and then approximate the rank with $(F-1)/N$, see Weston et al. (2011) for more details on this procedure. For the choices of the stochastic gradient learning rate, margin (γ) and dimensionality (m), please refer to §5.4-§5.5.

Note that an alternative approach could learn only the matrix M, and then use a k-nearest neighbor classifier in $\mathbb{R}^m$, as in Weinberger and Saul (2009). The advantage of learning an embedding for the frame labels is that at inference time we need to consider only the set of labels for classification rather than all training examples. Additionally, since we use a frame lexicon that gives us the possible frames for a given predicate, we usually only consider a handful of candidate labels. If we used all training examples for a given predicate for finding a nearest-neighbor match at inference time, we would have to consider many more candidates, making the process very slow.

4 Argument Identification

Here, we briefly describe the argument identification model used in our frame-semantic parsing experiments, post frame identification. Given x, the sentence with a marked predicate, the argument identification model assumes that the predicate frame y has been disambiguated. From a frame lexicon, we look up the set of semantic roles $\mathcal{R}_y$ that associate with y. This set also contains the null role $r_\emptyset$. From x, a rule-based candidate argument extraction algorithm extracts a set of spans $\mathcal{A}$ that could potentially serve as the overt[7] arguments $\mathcal{A}_y$ for y (see §5.4-§5.5 for the details of the candidate argument extraction algorithms).

[7]By overtness, we mean the non-null instantiation of a semantic role in a frame-semantic parse.

• starting word of a	• POS of the starting word of a
• ending word of a	• POS of the ending word of a
• head word of a	• POS of the head word of a
• bag of words in a	• bag of POS tags in a
• a bias feature	• voice of the predicate use
• word cluster of a's head	
• word cluster of a's head conjoined with word cluster of the predicate*	
• dependency path between a's head and the predicate	
• the set of dependency labels of the predicate's children	
• dependency path conjoined with the POS tag of a's head	
• dependency path conjoined with the word cluster of a's head	
• position of a with respect to the predicate (*before, after, overlap or identical*)	
• whether the subject of the predicate is missing (*missingsubj*)	
• *missingsubj*, conjoined with the dependency path	
• *missingsubj*, conjoined with the dependency path from the verb dominating the predicate to a's head	

Table 1: Argument identification features. The span in consideration is termed a. Every feature in this list has two versions, one conjoined with the given role r and the other conjoined with both r and the frame y. The feature with a * superscript is only conjoined with the role to reduce its sparsity.

Learning Given training data of the form $\langle\langle x^{(i)}, y^{(i)}, \mathcal{M}^{(i)}\rangle\rangle_{i=1}^{N}$, where,

$$\mathcal{M} = \{(r,a\} : r \in \mathcal{R}_y, a \in \mathcal{A} \cup \mathcal{A}_y\}, \quad (1)$$

a set of tuples that associates each role r in $\mathcal{R}_y$ with a span a according to the gold data. Note that this mapping associates spans with the null role $r_\emptyset$ as well. We optimize the following log-likelihood to train our model:

$$\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} \sum_{j=1}^{|\mathcal{M}^{(i)}|} \log p_{\boldsymbol{\theta}}\big((r,a)_j | x, y, \mathcal{R}_y\big) - C\|\boldsymbol{\theta}\|_2^2$$

where $p_{\boldsymbol{\theta}}$ is a log-linear model normalized over the set $\mathcal{R}_y$, with features described in Table 1. We set $C = 1.0$ and use L-BFGS (Liu and Nocedal, 1989) for training.

Inference Although our learning mechanism uses a local log-linear model, we perform inference globally on a per-frame basis by applying hard structural constraints. Following Das et al. (2014) and Punyakanok et al. (2008) we use the log-probability of the local classifiers as a score in an integer linear program (ILP) to assign roles subject to hard constraints described in §5.4 and §5.5. We use an off-the-shelf ILP solver for inference.

5 Experiments

In this section, we present our experiments and the results achieved. We evaluate our novel frame identification approach in isolation and also conjoined with argument identification resulting in full frame-semantic structures; before presenting our model's performance we first focus on the datasets, baselines and the experimental setup.

5.1 Data

We evaluate our models on both FrameNet- and PropBank-style structures. For FrameNet, we use the full-text annotations in the FrameNet 1.5 release[8] which was used by Das et al. (2014, §3.2). We used the same test set as Das et al. containing 23 documents with 4,458 predicates. Of the remaining 55 documents, 16 documents were randomly chosen for development.[9]

For experiments with PropBank, we used the Ontonotes corpus (Hovy et al., 2006), version 4.0, and only made use of the Wall Street Journal documents; we used sections 2-21 for training, section 24 for development and section 23 for testing. This resembles the setup used by Punyakanok et al. (2008). All the verb frame files in Ontonotes were used for creating our frame lexicon.

5.2 Frame Identification Baselines

For comparison, we implemented a set of baseline models, with varying feature configurations. The baselines use a log-linear model that models the following probability at training time:

$$p(y|x,\ell) = \frac{e^{\boldsymbol{\psi}\cdot\mathbf{f}(y,x,\ell)}}{\sum_{\bar{y}\in F_\ell} e^{\boldsymbol{\psi}\cdot\mathbf{f}(\bar{y},x,\ell)}} \quad (2)$$

At test time, this model chooses the best frame as $\text{argmax}_y \boldsymbol{\psi}\cdot\mathbf{f}(y,x,\ell)$ where argmax iterates over the possible frames $y \in F_\ell$ if ℓ was seen in the lexicon or the training data, or $y \in F$, if it was unseen, like the disambiguation scheme of §3. We train this model by maximizing L_2 regularized log-likelihood, using L-BFGS; the regularization constant was set to 0.1 in all experiments.

For comparison with our model from §3, which we call WSABIE EMBEDDING, we implemented two baselines with the log-linear model. Both the baselines use features very similar to the input representations described in §3.1. The first one computes the direct dependents and dependency paths as described in §3.1 but conjoins them with the word identity rather than a word embedding. Additionally, this model uses the un-conjoined words as backoff features. This would be a standard NLP approach for the frame identification problem, but is surprisingly competitive with the state of the art. We call this baseline LOG-LINEAR WORDS. The second baseline, tries to decouple the WSABIE training from the embedding input, and trains a log linear model using the embeddings. So the second baseline has the same input representation as WSABIE EMBEDDING but uses a log-linear model instead of WSABIE. We call this model LOG-LINEAR EMBEDDING.

5.3 Common Experimental Setup

We process our PropBank and FrameNet training, development and test corpora with a shift-reduce dependency parser that uses the Stanford conventions (de Marneffe and Manning, 2013) and uses an arc-eager transition system with beam size of 8; the parser and its features are described by Zhang and Nivre (2011). Before parsing the data, it is tagged with a POS tagger trained with a conditional random field (Lafferty et al., 2001) with the following emission features: word, the word cluster, word suffixes of length 1, 2 and 3, capitalization, whether it has a hyphen, digit and punctuation. Beyond the bias transition feature, we have two cluster features for the left and right words in the transition. We use Brown clusters learned using the algorithm of Uszkoreit and Brants (2008) on a large English newswire corpus for cluster features. We use the same word clusters for the argument identification features in Table 1.

We learn the initial embedding representations for our frame identification model (§3) using a deep neural language model similar to the one proposed by Bengio et al. (2003). We use 3 hidden layers each with 1024 neurons and learn a 128-dimensional embedding from a large corpus containing over 100 billion tokens. In order to speed up learning, we use an unnormalized output layer and a hinge-loss objective. The objective tries to ensure that the correct word scores higher than a random incorrect word, and we train with mini-batch stochastic gradient descent.

5.4 Experimental Setup for FrameNet

Hyperparameters For our frame identification model with embeddings, we search for the WSABIE hyperparameters using the development data.

[8] See `https://framenet.icsi.berkeley.edu`.

[9] These documents are listed in appendix A.

		SEMAFOR LEXICON			FULL LEXICON		
	Model	All	Ambiguous	Rare	All	Ambiguous	Rare
Development Data	LOG-LINEAR WORDS	96.21	90.41	95.75	96.37	90.41	96.07
	LOG-LINEAR EMBEDDING	96.06	90.56	95.38	96.19	90.49	95.70
	WSABIE EMBEDDING (§3)	**96.90**	**92.73**	**96.44**	**96.99**	**93.12**	**96.39**

		SEMAFOR LEXICON				FULL LEXICON		
	Model	All	Ambiguous	Rare	Unseen	All	Ambiguous	Rare
Test Data	Das et al. (2014) supervised	82.97	69.27	80.97	23.08			
	Das et al. (2014) best	83.60	69.19	82.31	42.67			
	LOG-LINEAR WORDS	84.71	70.97	81.70	27.27	87.44	70.97	87.10
	LOG-LINEAR EMBEDDING	83.42	68.70	80.95	27.97	86.20	68.70	86.03
	WSABIE EMBEDDING (§3)	**86.58**	**73.67**	**85.04**	**44.76**	**88.73**	**73.67**	**89.38**

Table 2: Frame identification results for FrameNet. See §5.6.

		SEMAFOR LEXICON			FULL LEXICON		
	Model	Precision	Recall	F_1	Precision	Recall	F_1
Development Data	LOG-LINEAR WORDS	89.43	75.98	82.16	89.41	76.05	82.19
	WSABIE EMBEDDING (§3)	**89.89**	**76.40**	**82.59**	**89.94**	**76.27**	**82.54**
Test Data	Das et al. supervised	67.81	60.68	64.05			
	Das et al. best	68.33	61.14	64.54			
	LOG-LINEAR WORDS	71.16	63.56	67.15	73.35	65.27	69.08
	WSABIE EMBEDDING (§3)	**72.79**	**64.95**	**68.64**	**74.44**	**66.17**	**70.06**

Table 3: Full structure prediction results for FrameNet; this reports frame and argument identification performance jointly. We skip LOG-LINEAR EMBEDDING because it underperforms all other models by a large margin.

We search for the stochastic gradient learning rate in $\{\underline{0.0001}, 0.001, 0.01\}$, the margin $\gamma \in \{0.001, \underline{0.01}, 0.1, 1\}$ and the dimensionality of the final vector space $m \in \{\underline{256}, 512\}$, to maximize the frame identification accuracy of *ambiguous* lexical units; by ambiguous, we imply lexical units that appear in the training data or the lexicon with more than one semantic frame. The underlined values are the chosen hyperparameters used to analyze the test data.

Argument Candidates The candidate argument extraction method used for the FrameNet data, (as mentioned in §4) was adapted from the algorithm of Xue and Palmer (2004) applied to dependency trees. Since the original algorithm was designed for verbs, we added a few extra rules to handle non-verbal predicates: we added 1) the predicate itself as a candidate argument, 2) the span ranging from the sentence position to the right of the predicate to the rightmost index of the subtree headed by the predicate's head; this helped capture cases like "a *few* months" (where *few* is the predicate and months is the argument), and 3) the span ranging from the leftmost index of the subtree headed by the predicate's head to the position immediately before the predicate, for cases like "your gift *to* Goodwill" (where *to* is the predicate and your gift is the argument).[10]

Frame Lexicon In our experimental setup, we scanned the XML files in the "frames" directory of the FrameNet 1.5 release, which lists all the frames, the corresponding roles and the associated lexical units, and created a frame lexicon to be used in our frame and argument identification models. We noted that this renders every lexical unit as *seen*; in other words, at frame disambiguation time on our test set, for all instances, we only had to score the frames in F_ℓ for a predicate with lexical unit ℓ (see §3 and §5.2). We call this setup FULL LEXICON. While comparing with prior state of the art on the same corpus, we noted that Das et al. (2014) found several unseen predicates at test time.[11] For fair comparison, we took the lexical units for the predicates that Das et al. considered as seen, and constructed a lexicon with only those; training instances, if any, for the unseen predicates under Das et al.'s setup were thrown out as well. We call this setup SEMAFOR LEXICON.[12] We also experimented on the set of unseen instances used by Das et al.

ILP constraints For FrameNet, we used three ILP constraints during argument identification (§4). 1) each span could have only one role, 2) each core role could be present only once, and 3) all overt arguments had to be non-overlapping.

[10] Note that Das et al. (2014) describe the state of the art in FrameNet-based analysis, but their argument identification strategy considered all possible dependency subtrees in a parse, resulting in a much larger search space.

[11] Instead of using the frame files, Das et al. built a frame lexicon from FrameNet's exemplars and the training corpus.

[12] We got Das et al.'s seen predicates from the authors.

Model	All	Ambiguous	Rare
LOG-LINEAR WORDS	94.21	90.54	93.33
LOG-LINEAR EMBEDDING	93.81	89.86	**93.73**
WSABIE EMBEDDING (§3)	**94.79**	**91.52**	92.55
Dev data ↑ ↓ Test data			
Model	All	Ambiguous	Rare
LOG-LINEAR WORDS	**94.74**	**92.07**	**91.32**
LOG-LINEAR EMBEDDING	94.04	90.95	90.97
WSABIE EMBEDDING (§3)	94.56	91.82	90.62

Table 4: Frame identification accuracy results for PropBank. The model and the column names have the same semantics as Table 2.

Model	P	R	F_1
LOG-LINEAR WORDS	80.02	75.58	77.74
WSABIE EMBEDDING (§3)	**80.06**	**75.74**	**77.84**
Dev data ↑ ↓ Test data			
Model	P	R	F_1
LOG-LINEAR WORDS	**81.55**	**77.83**	**79.65**
WSABIE EMBEDDING (§3)	81.32	77.97	79.61

Table 5: Full frame-structure prediction results for Propbank. This is a metric that takes into account frames and arguments together. See §5.7 for more details.

5.5 Experimental Setup for PropBank

Hyperparameters As in §5.4, we made a hyperparameter sweep in the same space. The chosen learning rate was 0.01, while the other values were $\gamma = 0.01$ and $m = 512$. Ambiguous lexical units were used for this selection process.

Argument Candidates For PropBank we use the algorithm of Xue and Palmer (2004) applied to dependency trees.

Frame Lexicon For the PropBank experiments we scanned the frame files for propositions in Ontonotes 4.0, and stored possible core roles for each verb frame. The lexical units were simply the verb associating with the verb frames. There were no unseen verbs at test time.

ILP constraints We used the constraints of Punyakanok et al. (2008).

5.6 FrameNet Results

Table 2 presents accuracy results on frame identification.[13] We present results on all predicates, ambiguous predicates seen in the lexicon or the training data, and rare ambiguous predicates that appear ≤ 11 times in the training data. The WSABIE EMBEDDING model from §3 performs significantly better than the LOG-LINEAR WORDS baseline, while LOG-LINEAR EMBEDDING underperforms in every metric. For the SEMAFOR LEXICON setup, we also compare with the state of the art from Das et al. (2014), who used a semi-supervised learning method to improve upon a supervised latent-variable log-linear model. For unseen predicates from the Das et al. system, we perform better as well. Finally, for the FULL LEXICON setting, the absolute accuracy numbers are even better for our best model. Table 3 presents results on the full frame-semantic parsing task (measured by a reimplementation of the SemEval 2007 shared task evaluation script) when our argument identification model (§4) is used after frame identification. We notice similar trends as in Table 2, and our results outperform the previously published best results, setting a new state of the art.

[13] We do not report partial frame accuracy that has been reported by prior work.

Model	P	R	F_1
LOG-LINEAR WORDS	**77.29**	**71.50**	**74.28**
WSABIE EMBEDDING (§3)	77.13	71.32	74.11
Dev data ↑ ↓ Test data			
Model	P	R	F_1
LOG-LINEAR WORDS	**79.47**	**75.11**	**77.23**
WSABIE EMBEDDING (§3)	79.36	75.04	77.14
Punyakanok et al. *Collins*	75.92	71.45	73.62
Punyakanok et al. *Charniak*	77.09	75.51	76.29
Punyakanok et al. *Combined*	80.53	76.94	78.69

Table 6: Argument only evaluation (semantic role labeling metrics) using the CoNLL 2005 shared task evaluation script (Carreras and Màrquez, 2005). Results from Punyakanok et al. (2008) are taken from Table 11 of that paper.

5.7 PropBank Results

Table 4 shows frame identification results on the PropBank data. On the development set, our best model performs with the highest accuracy on all and ambiguous predicates, but performs worse on rare ambiguous predicates. On the test set, the LOG-LINEAR WORDS baseline performs best by a very narrow margin. See §6 for a discussion.

Table 5 presents results where we measure precision, recall and F_1 for frames and arguments together; this strict metric penalizes arguments for mismatched frames, like in Table 3. We see the same trend as in Table 4. Finally, Table 6 presents SRL results that measures argument performance only, irrespective of the frame; we use the evaluation script from CoNLL 2005 (Carreras and Màrquez, 2005). We note that with a better frame identification model, our performance on SRL improves in general. Here, too, the embedding model barely misses the performance of the best baseline, but we are at par and sometimes better than the single parser setting of a state-of-the-art SRL system (Punyakanok et al., 2008).[14]

[14] The last row of Table 6 refers to a system which used the

6 Discussion

For FrameNet, the WSABIE EMBEDDING model we propose strongly outperforms the baselines on all metrics, and sets a new state of the art. We believe that the WSABIE EMBEDDING model performs better than the LOG-LINEAR EMBEDDING baseline (that uses the same input representation) because the former setting allows examples with different labels and confusion sets to share information; this is due to the fact that all labels live in the same label space, and a single projection matrix is shared across the examples to map the input features to this space. Consequently, the WSABIE EMBEDDING model can share more information between different examples in the training data than the LOG-LINEAR EMBEDDING model. Since the LOG-LINEAR WORDS model always performs better than the LOG-LINEAR EMBEDDING model, we conclude that the primary benefit does not come from the input embedding representation.[15]

On the PropBank data, we see that the LOG-LINEAR WORDS baseline has roughly the same performance as our model on most metrics: slightly better on the test data and slightly worse on the development data. This can be partially explained with the significantly larger training set size for PropBank, making features based on words more useful. Another important distinction between PropBank and FrameNet is that the latter shares frames between multiple lexical units. The effect of this is clearly observable from the "Rare" column in Table 4. WSABIE EMBEDDING performs poorly in this setting while LOG-LINEAR EMBEDDING performs well. Part of the explanation has to do with the specifics of WSABIE training. Recall that the WSABIE EMBEDDING model needs to estimate the label location in $\mathbb{R}^m$ for each frame. In other words, it must estimate 512 parameters based on at most 10 training examples. However, since the input representation is shared across all frames, every other training example from all the lexical units affects the optimal estimate, since they all modify the joint parameter matrix M. By contrast, in the log-linear models each label has its own set of parameters, and they interact only via the normalization constant. The LOG-LINEAR WORDS model does not have this entanglement, but cannot share information between words. For PropBank, these drawbacks and benefits balance out and we see similar performance for LOG-LINEAR WORDS and LOG-LINEAR EMBEDDING. For FrameNet, estimating the label embedding is not as much of a problem because even if a lexical unit is rare, the potential frames can be frequent. For example, we might have seen the SENDING frame many times, even though *telex*.V is a rare lexical unit.

In comparison to prior work on FrameNet, even our baseline models outperform the previous state of the art. A particularly interesting comparison is between our LOG-LINEAR WORDS baseline and the supervised model of Das et al. (2014). They also use a log-linear model, but they incorporate a latent variable that uses WordNet (Fellbaum, 1998) to get lexical-semantic relationships and smooths over frames for ambiguous lexical units. It is possible that this reduces the model's power and causes it to over-generalize. Another difference is that when training the log-linear model, they normalize over all frames, while we normalize over the allowed frames for the current lexical unit. This would tend to encourage their model to expend more of its modeling power to rule out possibilities that will be pruned out at test time.

7 Conclusion

We have presented a simple model that outperforms the prior state of the art on FrameNet-style frame-semantic parsing, and performs at par with one of the previous-best single-parser systems on PropBank SRL. Unlike Das et al. (2014), our model does not rely on heuristics to construct a similarity graph and leverage WordNet; hence, in principle it is generalizable to varying domains, and to other languages. Finally, we presented results on PropBank-style semantic role labeling with a system that included the task of automatic verb frame identification, in tune with the FrameNet literature; we believe that such a system produces more interpretable output, both from the perspective of human understanding as well as downstream applications, than pipelines that are oblivious to the verb frame, only focusing on argument analysis.

Acknowledgments

We thank Emily Pitler for comments on an early draft, and the anonymous reviewers for their valuable feedback.

combination of two syntactic parsers as input.

[15] One could imagine training a WSABIE model with word features, but we did not perform this experiment.

References

C. F. Baker, C. J. Fillmore, and J. B. Lowe. 1998. The berkeley framenet project. In *Proceedings of COLING-ACL*.

C. Baker, M. Ellsworth, and K. Erk. 2007. SemEval-2007 Task 19: Frame semantic structure extraction. In *Proceedings of SemEval*.

Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

X. Carreras and L. Màrquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of CoNLL*.

X. Carreras and L. Màrquez. 2005. Introduction to the CoNLL-2005 shared task: semantic role labeling. In *Proceedings of CoNLL*.

R. Collobert and J. Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*.

D. Das, N. Schneider, D. Chen, and N. A. Smith. 2010. Probabilistic frame-semantic parsing. In *Proceedings of NAACL-HLT*.

D. Das, D. Chen, A. F. T. Martins, N. Schneider, and N. A. Smith. 2014. Frame-semantic parsing. *Computational Linguistics*, 40(1):9–56.

M.-C. de Marneffe and C. D. Manning, 2013. *Stanford typed dependencies manual*.

C. Fellbaum, editor. 1998. *WordNet: an electronic lexical database*.

C. J. Fillmore, C. R. Johnson, and M. R. Petruck. 2003. Background to FrameNet. *International Journal of Lexicography*, 16(3):235–250.

C. J. Fillmore. 1982. Frame Semantics. In *Linguistics in the Morning Calm*, pages 111–137. Hanshin Publishing Co., Seoul, South Korea.

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

E. Hovy, M. Marcus, M. Palmer, L. Ramshaw, and R. Weischedel. 2006. Ontonotes: The 90 In *Proceedings of NAACL-HLT*.

R. Johansson and P. Nugues. 2007. LTH: semantic structure extraction using nonprojective dependency trees. In *Proceedings of SemEval*.

A. Klementiev, I. Titov, and B. Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *Proceedings of COLING*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*.

D. C. Liu and J. Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(3):503 – 528.

L. Màrquez, X. Carreras, K. C. Litkowski, and S. Stevenson. 2008. Semantic role labeling: an introduction to the special issue. *Computational Linguistics*, 34(2):145–159.

A. Meyers, R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Young, and R. Grishman. 2004. The NomBank project: An interim report. In *Proceedings of NAACL/HLT Workshop on Frontiers in Corpus Annotation*.

J. Mitchell and M. Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL-HLT*.

M. Palmer, D. Gildea, and P. Kingsbury. 2005. The Proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

V. Punyakanok, D. Roth, and W. Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of EMNLP*.

J. Turian, L. Ratinov, and Y. Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of ACL*, Stroudsburg, PA, USA.

N. Usunier, D. Buffoni, and P. Gallinari. 2009. Ranking with ordered weighted pairwise classification. In *ICML*.

J. Uszkoreit and T. Brants. 2008. Distributed word clustering for large scale class-based language modeling in machine translation. In *Proceedings of ACL-HLT*.

K. Q. Weinberger and L. K. Saul. 2009. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244.

J. Weston, S. Bengio, and N. Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *Proceedings of IJCAI*.

N. Xue and M. Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of EMNLP 2004*.

Y. Zhang and J. Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of ACL-HLT*.

Number	Filename
dev-1	LUCorpus-v0.3__20000420_xin_eng-NEW.xml
dev-2	NTI__SouthAfrica_Introduction.xml
dev-3	LUCorpus-v0.3__CNN_AARONBROWN_ENG_20051101_215800.partial-NEW.xml
dev-4	LUCorpus-v0.3__AFGP-2002-600045-Trans.xml
dev-5	PropBank__TicketSplitting.xml
dev-6	Miscellaneous__Hijack.xml
dev-7	LUCorpus-v0.3__artb_004_A1_E1_NEW.xml
dev-8	NTI__WMDNews_042106.xml
dev-9	C-4__C-4Text.xml
dev-10	ANC__EntrepreneurAsMadonna.xml
dev-11	NTI__LibyaCountry1.xml
dev-12	NTI__NorthKorea_NuclearOverview.xml
dev-13	LUCorpus-v0.3__20000424_nyt-NEW.xml
dev-14	NTI__WMDNews_062606.xml
dev-15	ANC__110CYL070.xml
dev-16	LUCorpus-v0.3__CNN_ENG_20030614_173123.4-NEW-1.xml

Table 7: List of files used as development set for the FrameNet 1.5 corpus.

A Development Data

Table 7 features a list of the 16 randomly selected documents from the FrameNet 1.5 corpus, which we used for development. The resultant development set consists of roughly 4,500 predicates. We use the same test set as in Das et al. (2014), containing 23 documents and 4,458 predicates.

A Sense-Based Translation Model for Statistical Machine Translation

Deyi Xiong and **Min Zhang***
Provincial Key Laboratory for Computer Information Processing Technology
Soochow University, Suzhou, China 215006
{dyxiong, minzhang}@suda.edu.cn

Abstract

The sense in which a word is used determines the translation of the word. In this paper, we propose a sense-based translation model to integrate word senses into statistical machine translation. We build a broad-coverage sense tagger based on a nonparametric Bayesian topic model that automatically learns sense clusters for words in the source language. The proposed sense-based translation model enables the decoder to select appropriate translations for source words according to the inferred senses for these words using maximum entropy classifiers. Our method is significantly different from previous word sense disambiguation reformulated for machine translation in that the latter neglects word senses in nature. We test the effectiveness of the proposed sense-based translation model on a large-scale Chinese-to-English translation task. Results show that the proposed model substantially outperforms not only the baseline but also the previous reformulated word sense disambiguation.

1 Introduction

One of very common phenomena in language is that a plenty of words have multiple meanings. In the context of machine translation, such different meanings normally produce different target translations. Therefore a natural assumption is that word sense disambiguation (WSD) may contribute to statistical machine translation (SMT) by providing appropriate word senses for target translation selection with context features (Carpuat and Wu, 2005).

This assumption, however, has not been empirically verified in the early days. Carpuat and Wu (2005) adopt a standard formulation of WSD: predicting word senses that are defined on an ontology for ambiguous words. As they apply WSD to Chinese-to-English translation, they predict word senses from a Chinese ontology HowNet and project the predicted senses to English glosses provided by HowNet. These glosses, used as the sense predictions of their WSD system, are integrated into a word-based SMT system either to substitute for translation candidates of their translation model or to postedit the output of their SMT system. They report that WSD degenerates the translation quality of SMT.

In contrast to the standard WSD formulation, Vickrey et al. (2005) reformulate the task of WSD for SMT as predicting possible target translations rather than senses for ambiguous source words. They show that such a reformulated WSD can improve the accuracy of a simplified word translation task.

Following this WSD reformulation for SMT, Chan et al. (2007) integrate a state-of-the-art WSD system into a hierarchical phrase-based system (Chiang, 2005). Carpuat and Wu (2007) also use this reformulated WSD and further adapt it to multi-word phrasal disambiguation. They both report that the redefined WSD can significantly improve SMT.

Although this reformulated WSD has proved helpful for SMT, one question is not answered yet: are pure word senses useful for SMT? The early WSD for SMT (Carpuat and Wu, 2005) uses projected word senses while the reformulated WSD sidesteps word senses. In this paper we would like to re-investigate this question by resorting to word sense induction (WSI) that is related to but different from WSD.[1] We use

*Corresponding author

[1] We will discuss the relation and difference between WSI and WSD in Section 2.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1459–1469,
Baltimore, Maryland, USA, June 23-25 2014.

WSI to obtain word senses for large-scale data. With these word senses, we study in particular: 1) whether word senses can be directly integrated to SMT to improve translation quality and 2) whether WSI-based model can outperform the reformulated WSD in the context of SMT.

In order to incorporate word senses into SMT, we propose a sense-based translation model that is built on maximum entropy classifiers. We use a nonparametric Bayesian topic model based WSI to infer word senses for source words in our training, development and test set. We collect training instances from the sense-tagged training data to train the proposed sense-based translation model. Specially,

- Instead of predicting target translations for ambiguous source words as the previous reformulated WSD does, we first predict word senses for ambiguous source words. The predicted word senses together with other context features are then used to predict possible target translations for these words.

- Instead of using word senses defined by a prespecified sense inventory as the standard WSD does, we incorporate word senses that are automatically learned from data into our sense-based translation model.

We integrate the proposed sense-based translation model into a state-of-the-art SMT system and conduct experiments on Chines-to-English translation using large-scale training data. Results show that automatically learned word senses are able to improve translation quality and the sense-based translation model is better than the previous reformulated WSD.

The remainder of this paper proceeds as follows. Section 2 introduces how we obtain word senses for our large-scale training data via a WSI-based broad-coverage sense tagger. Section 3 presents our sense-based translation model. Section 4 describes how we integrate the sense-based translation model into SMT. Section 5 elaborates our experiments on the large-scale Chinese-to-English translation task. Section 6 introduces related studies and highlights significant differences from them. Finally, we conclude in Section 7 with future directions.

2 WSI-Based Broad-Coverage Sense Tagger

In order to obtain word senses for any source words, we build a broad-coverage sense tagger that relies on the nonparametric Bayesian model based word sense induction. We first describe WSI, especially WSI based on the Hierarchical Dirichlet Process (HDP) (Teh et al., 2004), a nonparametric version of Latent Dirichlet Allocation (LDA) (Blei et al., 2003). We then elaborate how we use the HDP-based WSI to predict sense clusters and to annotate source words in our training/development/test sets with these sense clusters.

2.1 Word Sense Induction

Before we introduce WSI, we differentiate **word type** from **word token**. A word type refers to a unique word as a vocabulary entry while a word token is an occurrence of a word type. Take the first sentence of this paragraph as an example, it has 11 word tokens but 9 word types as there are two word tokens of the word type "we" and two tokens of the word type "word".

Word sense induction is a task of automatically inducing the underlying senses of word tokens given the surrounding contexts where the word tokens occur. The biggest difference from word sense disambiguation lies in that WSI does not rely on a predefined sense inventory. Such a prespecified list of senses is normally assumed by WSD which predicts senses of word tokens using this given inventory. From this perspective, WSI can be treated as a clustering problem while WSD a classification one.

Various clustering algorithms, such as k-means, have been previously used for WSI. Recently, we have also witnessed that WSI is cast as a topic modeling problem where the sense clusters of a word type are considered as underlying topics (Brody and Lapata, 2009; Yao and Durme, 2011; Lau et al., 2012). We follow this line to tailor a topic modeling framework to induce word senses for our large-scale training data.

In the topic-based WSI, surrounding context of a word token is considered as a **pseudo document** of the corresponding word type. A pseudo document is composed of either a bag of neighboring words of a word token, or the Part-to-Speech tags of neighboring words, or other contextual information elements. In this paper, we define a pseudo

document as $\pm N$ neighboring words centered on a given word token. Table 1 shows examples of pseudo documents for a Chinese word "wǎngluò" (network). These two pseudo documents are extracted from a sentence listed in the first row of Table 1. Here we set $N = 5$. We can extract as many pseudo documents as the number of word tokens of a given word type that occur in training data. The collection of all these extracted pseudo documents of the given word type forms a corpus. We can induce topics on this corpus for each pseudo document via topic modeling approaches.

Figure 1(a) shows the LDA-based WSI for a given word type W. The outer plate represents replicates of pseudo documents which consist of N neighboring words centered on the tokens of the given word type W. $w_{j,i}$ is the i-th word of the j-th pseudo document of the given word type W. $s_{j,i}$ is the sense assigned to the word $w_{j,i}$. The conventional topic distribution θ_j for the j-th pseudo document is taken as the the distribution over senses for the given word type W. The LDA generative process for sense induction is as follows: 1) for each pseudo document D_j, draw a per-document sense distribution θ_j from a Dirichlet distribution Dir(α); 2) for each item $w_{j,i}$ in the pseudo document D_j, 2.1) draw a sense cluster $s_{j,i} \sim$ Multinomial(θ_j); and 2.2) draw a word $w_{j,i} \sim \varphi_{s_{j,i}}$ where $\varphi_{s_{j,i}}$ is the distribution of sense $s_{j,i}$ over words drawn from a Dirichlet distribution Dir(β).

As LDA needs to manually specify the number of senses (topics), a better idea is to let the training data automatically determine the number of senses for each word type. Therefore we resort to the HDP, a natural nonparametric generalization of LDA, for the inference of both sense clusters and the number of sense clusters following Lau et al. (2012) and Yao and Durme (2011). The HDP for WSI is shown in Figure 1(b). The HDP generative process for word sense induction is as follows: 1) sample a base distribution G_0 from a Dirichlet process DP(γ, H) with a concentration parameter γ and a base distribution H; 2) for each pseudo document D_j, sample a distribution $G_j \sim$ DP(α_0, G_0); 3) for each item $w_{j,i}$ in the pseudo document D_j, 3.1) sample a sense cluster $s_{j,i} \sim G_j$; and 3.2) sample a word $w_{j,i} \sim \varphi_{s_{j,i}}$. Here G_0 is a global distribution over sense clusters that are shared by all G_j. G_j is a per-document sense distribution over these sense clusters, which has its own document-specific proportions of these sense clusters. The hyperparameter γ, α_0 in the HDP are both concentration parameters which control the variability of senses in the global distribution G_0 and document-specific distribution G_j.

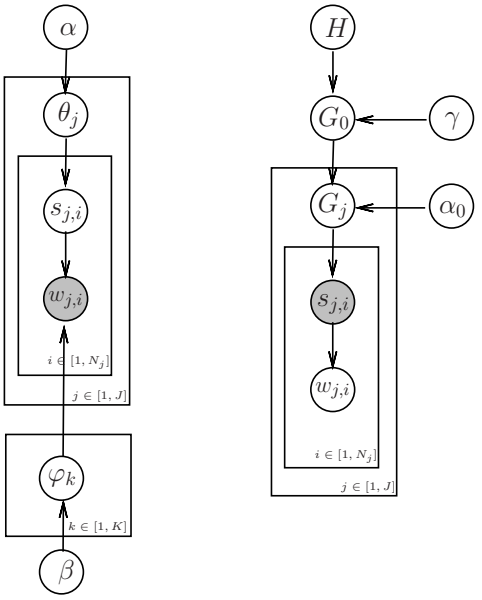


Figure 1: Graphical model representations of (a) Latent Dirichlet Allocation for WSI, (b) Hierarchical Dirichlet Process for WSI.

The HDP/LDA-based WSI complies with the distributional hypothesis that states that words occurring in the same contexts tend to have similar meanings. We want to extend this hypothesis to machine translation by building sense-based translation model upon the HDP-based word sense induction: words with the same meanings tend to be translated in the same way.

2.2 Word Sense Tagging

We adopt the HDP-based WSI to automatically predict word senses and use these predicted senses to annotate source words. We individually build a HDP-based WSI model per word type and train these models on the training data. The sense for a word token is defined as the most probable sense according to the per-document sense distribution G_j estimated for the corresponding pseudo document that represents the surrounding context of the word token. In particular, we take the following steps.

tā tíxǐng wǒguó wǎngluò yùnyíng zhě zhùyì fángfàn hēikè gōngjī ， quèbǎo wǎngluò ānquán 。
Pseudo Documents for word “wǎngluò”
tā tíxǐng wǒguó wǎngluò yùnyíng zhě zhùyì fángfàn hēikè fángfàn hēikè gōngjī ， quèbǎo wǎngluò ānquán 。

Table 1: Examples of pseudo documents extracted from a Chinese sentence (written in Chinese Pinyin).

- **Data preprocessing** We preprocess the source side of our bilingual training data as well as development and test set by removing stop words and rare words.
- **Training Data Sense Annotation** From the preprocessed training data, we extract all possible pseudo documents for each source word type. The collection of these extracted pseudo documents is used as a corpus to train a HDP-based WSI model for the source word type. In this way, we can train as many HDP-based WSI models as the number of word types kept after preprocessing. The sense with the highest probability output by the HDP-based WSI model for each pseudo document is used as the sense cluster to label the corresponding word token.
- **Test/Dev Data Sense Annotation** From the preprocessed test data, we can also extract pseudo documents for each source word type that occur in the test/dev set. Using the trained HDP-based WSI model that correspond to the source word type in question, we can obtain the best sense assignment for each pseudo document of the word type, which in turn is used to annotate the corresponding word token in the test/dev data.

3 Sense-Based Translation Model

In this section we present our sense-based translation model and describe the features that we use as well as the training process of this model.

3.1 Model

The sense-based translation model estimates the probability that a source word c is translated into a target phrase $\tilde{e}$ given contextual information, including word senses that are obtained using the HDP-based WSI as described in the last section. We allow the target phrase $\tilde{e}$ to be either a phrase of length up to 3 words or NULL so that we can capture both multi-word and null translations. The essential component of the model is a maximum entropy (MaxEnt) based classifier that is used to predict the translation probability $p(\tilde{e}|\mathcal{C}(c))$. The MaxEnt classifier can be formulated as follows.

$$p(\tilde{e}|\mathcal{C}(c)) = \frac{exp(\sum_i \theta_i h_i(\tilde{e}, \mathcal{C}(c)))}{\sum_{\tilde{e}'} exp(\sum_i \theta_i h_i(\tilde{e}', \mathcal{C}(c)))} \quad (1)$$

where h_is are binary features, θ_is are weights of these features, $\mathcal{C}(c)$ is the surrounding context of c.

We define two groups of binary features: 1) **lexicon features** and 2) **sense features**. All these features take the following form.

$$h(\tilde{e}, \mathcal{C}(c)) = \begin{cases} 1, & \text{if } \tilde{e} = \square \text{ and } \mathcal{C}(c).\mu = \nu \\ 0, & \text{else} \end{cases} \quad (2)$$

where $\square$ is a placeholder for a possible target translation (up to 3 words or NULL), μ is the name of a contextual (lexicon or sense) feature for the source word c, and the symbol ν represents the value of the feature μ.

We extract both the lexicon and sense features from a $\pm k$-word window centered on the word c. The lexicon features are defined as the preceding k words, the succeeding k words and the word c itself: $\{c_{-k}, ..., c_{-1}, c, c_1, ..., c_k\}$. The sense features are defined as the predicted senses for these words: $\{s_{c_{-k}}, ..., s_{c_{-1}}, s_c, s_{c_1}, ..., s_{c_k}\}$.

As we also use these neighboring words to predict word senses in the HDP-based WSI, the information provided by the lexicon and sense features may overlap. This is not a issue for the MaxEnt classifier as it can deal with arbitrary overlapping features (Berger et al., 1996). One may also wonder whether the sense features can contribute to SMT new information that can NOT be obtained from the lexicon features. First, we believe that the senses induced by the HDP-based WSI provide a different view of data than that of the lexicon features. Second, the sense features contain semantic distributional information learned by the HDP across contexts where lexical words occur. Third, we empirically investigate this doubt by comparing two MaxEnt-based translation models

in Section 5. One model only uses the lexicon features while the other integrates both the lexicon and sense features. The former model can be considered as a reformulated WSD for SMT as we described in Section 1.

Given a source sentence $\{c_i\}_1^I$, the proposed sense-based translation model M_s can be denoted as

$$M_s = \prod_{c_i \in \mathcal{W}} (\tilde{e_i}|\mathcal{C}(c_i)) \quad (3)$$

where $\mathcal{W}$ is a set of words for which we build MaxEnt classifiers (see the next subsection for the discussion on how we build MaxEnt classifiers for our sense-based translation model).

3.2 Training

The training of the proposed sense-based translation model is a process of estimating the feature weights θs in the equation (1). There are two strategies that we can use to obtain these weights. We can either build an all-in-one MaxEnt classifier that integrates all source word types c and their possible target translations $\tilde{e}$ or build multiple MaxEnt classifiers. If we train the all-in-one classifier, we have to predict millions of classes (target translations of length up to 3 words). This is normally intractable in practice. Therefore we take the second strategy: building multiple MaxEnt classifiers with one classifier per source word type.

In order to train these classifiers, we have to collect training events from our word-aligned bilingual training data where source words are annotated with their corresponding sense clusters predicted by the HDP-based WSI as described in Section 2. A training event for a source word c consists of all contextual elements in the form of $\mathcal{C}(c).\mu = \nu$ defined in the last subsection and the target translation $\tilde{e}$. Using these collected events, we can train our multiple classifiers. In practice, we do not build MaxEnt classifiers for source words that occur less than 10 times in the training data and run the MaxEnt toolkit in a parallel manner in order to expedite the training process.

4 Decoding with Sense-Based Translation Model

The sense-based translation model described above is integrated into the log-linear translation model of SMT as a sense-based knowledge source. The weight of this model is tuned by the minimum error rate training (MERT) (Och, 2003) together with other models such as the language model.

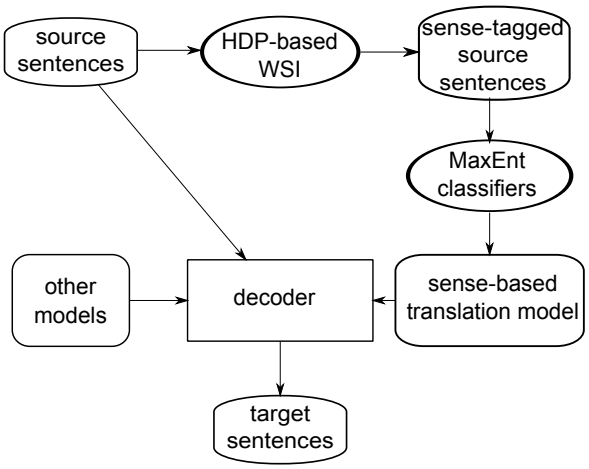


Figure 2: Architecture of SMT system with the sense-based translation model.

Figure 2 shows the architecture of the SMT system enhanced with the sense-based translation model. Before we translate a source sentence, we use the HDP-based WSI models trained on the training data to predict senses for word tokens occurring in the source sentence as discussed in Section 2.2. Note that the HDP-based WSI does not predict senses for all words due to the following two reasons.

- We do not train HDP-based WSI models for word types for which we extract more than T pseudo documents.[2]
- In the test/dev set, there are some words that are unseen in the training data. These unseen words, of course, do not have their HDP-based WSI models.

For these words, we set a default sense (i.e. $s_c = s_1$).

Sense tagging on test sentences can be done in a preprocessing step. Once we get sense clusters for word tokens in test sentences, we load pre-trained MaxEnt classifiers of the corresponding word types. During decoding, we keep word alignments for each translation rule. Whenever a new source word c is translated, we find its translation $\tilde{e}$ via the kept word alignments. We then calculate the translation probability $p(\tilde{e}|\mathcal{C}(c))$ according to the equation (1) using the corresponding loaded classifier. In this way, we can easily calculate the sense-based translation model score.

[2] we set $T = 20,000$.

5 Experiments

In this section, we carried out a series of experiments on Chinese-to-English translation using large-scale bilingual training data. In order to build the proposed sense-based translation model, we annotated the source part of the bilingual training data with word senses induced by the HDP-based WSI. With the trained sense-based translation model, we would like to investigate the following two questions:

- Do word senses automatically induced by the HDP-based WSI improve translation quality?
- Does the sense-based translation model outperform the reformulated WSD for SMT?

5.1 Setup

Our baseline system is a state-of-the-art SMT system which adapts Bracketing Transduction Grammars (Wu, 1997) to phrasal translation and equips itself with a maximum entropy based reordering model (Xiong et al., 2006). We used LDC corpora LDC2004E12, LDC2004T08, LDC2005T10, LDC2003E14, LDC2002E18, LDC2005T06, LDC2003E07, LDC2004T07 as our bilingual training data which consists of 3.84M bilingual sentences, 109.5M English word tokens and 96.9M Chinese word tokens. We ran Giza++ on the training data in two directions and applied the "grow-diag-final" refinement rule (Koehn et al., 2003) to obtain word alignments. From the word-aligned data, we extracted weighted phrase pairs to generate our phrase table. We trained a 5-gram language model on the Xinhua section of the English Gigaword corpus (306 million words) using the SRILM toolkit (Stolcke, 2002) with the modified Kneser-Ney smoothing (Chen and Goodman, 1996).

We trained our HDP-based WSI models via the C++ HDP toolkit[3] (Wang and Blei, 2012). We set the hyperparameters $\gamma = 0.1$ and $\alpha_0 = 1.0$ following Lau et al. (2012).We extracted pseudo documents from a $\pm$10-word window centered on the corresponding word token for each word type following Brody and Lapata (2009). As described in Section 2.2, we preprocessed the source part of our bilingual training data by removing stop words and infrequent words that occurs less than 10 times in the training data. From the preprocessed data, we extracted pseudo documents for each word type to train a HDP-based WSI model per word type. Note that we do not build WSI models for highly frequent words that occur more than 20,000 times in order to expedite the HDP training process.

We trained our MaxEnt classifiers with the off-the-shelf MaxEnt tool.[4] We performed 100 iterations of the L-BFGS algorithm implemented in the training toolkit on the collected training events from the sense-annotated data as described in Section 3.2. We set the Gaussian prior to 1 to avoid overfitting. On average, we obtained 346 classes (target translations) per source word type with the maximum number of classes being 256,243. It took an average of 57.5 seconds for training a Maxent classifier.

We used the NIST MT03 evaluation test data as our development set, and the NIST MT05 as the test set. We evaluated translation quality with the case-insensitive BLEU-4 (Papineni et al., 2002) and NIST (Doddington, 2002). In order to alleviate the impact of MERT (Och, 2003) instability, we followed the suggestion of Clark et al. (2011) to run MERT three times and report average BLEU/NIST scores over the three runs for all our experiments.

	Training	Test
# Word Types	67,723	4,348
# Total Pseudo Documents	27.73M	11,777
# Avg Pseudo Documents	427.79	2.71
# Total Senses	271,770	24,162
# Avg Senses	4.01	5.56

Table 2: Statistics of the HDP-based word sense induction on the training and test data.

5.2 Statistics and Examples of Word Senses

Before we present our experiment results of the sense-based translation model, we study some statistics of the HDP-based WSI on the training and test data. We show these statistics in Table 2. There are 67,723 and 4,348 unique word types in the training and test data after the preprocessing step. For these word types, we extract 27.73M and 11,777 pseudo documents from the training and test set respectively. On average, there are 427.79

[3] http://www.cs.cmu.edu/~chongw/resource.html

[4] http://homepages.inf.ed.ac.uk/lzhang10/maxenttoolkit.html

System	BLEU(%)	NIST
STM (±5w)	34.64	9.4346
STM (±10w)	**34.76**	**9.5114**
STM (±15w)	-	-

Table 4: Experiment results of the sense-based translation model (STM) with lexicon and sense features extracted from a window of size varying from ±5 to ±15 words on the development set.

pseudo documents per word type in the training data and 2.71 in the test set. The HDP-based WSI learns 271,770 word senses in total using the pseudo documents collected from the training data and infers 24,162 word senses using the pseudo documents extracted from the test set. There are 4.01 different senses per word type in the training data and 5.56 in the test set on average.

Table 3 illustrates six different senses of the word "运营 (operate)" learned by the HDP-based WSI in the training data. We also show the most probable 10 words for each sense cluster. Sense s_1 represents the operations of company or organization, sense s_2 denotes country/institution/internation operations, sense s_3 refers to market operations, sense s_4 corresponds to business operations, sense s_5 to public facility operations, and finally s_6 to economy operations.

5.3 Impact of Window Size k used in MaxEnt Classifiers

Our first group of experiments were conducted to investigate the impact of the window size k on translation performance in terms of BLEU/NIST on the development set. We extracted both the lexicon and sense features from a $\pm k$-word window for our MaxEnt classifiers. We varied k from 5 to 15. Experiment results are shown in Table 4. We achieve the best performance when $k = 10$. This suggests that a ±10-word window context is sufficient for predicting target translations for ambiguous source words. We therefore set $k = 10$ for all experiments thereafter.

5.4 Effect of the Sense-Based Translation Model

Our second group of experiments were carried out to investigate whether the sense-base translation model is able to improve translation quality by comparing the system enhanced with our sense-based translation model against the baseline. We also studied the impact of word senses induced by the HDP-based WSI on translation performance by enforcing the sense-based translation model to use only sense features. Table 5 shows the experiment results. From the table, we can observe that

System	BLEU(%)	NIST
Base	33.53	9.0561
STM (sense)	34.15	9.2596
STM (sense+lexicon)	34.73	9.4184

Table 5: Experiment results of the sense-based translation model (STM) against the baseline.

System	BLEU(%)	NIST
Base	33.53	9.0561
Reformulated WSD	34.16	9.3820
STM	34.73	9.4184

Table 6: Comparison results of the sense-based translation model vs. the reformulated WSD for SMT.

- Our sense-based translation model achieves a substantial improvement of 1.2 BLEU points over the baseline. This indicates that the sense-based translation model is able to help select correct translations for ambiguous source words.
- If we only integrate sense features into the sense-based translation model, we can still outperform the baseline by 0.62 BLEU points. This suggests that automatically induced word senses alone are indeed useful for machine translation.

5.5 Comparison to Word Sense Disambiguation

As we mentioned in Section 3.1, our sense-based translation model can be degenerated to a reformulated WSD model for SMT if we only use lexicon features in MaxEnt classifiers. This allows us to directly compare our method against the reformulated WSD for SMT. Table 6 shows the comparison result.

From the table, we can find that the sense-based translation model outperforms the reformulated WSD by 0.57 BLEU points. This suggests that the HDP-based word sense induction is better than the reformulated WSD in the context of SMT. Furthermore, as the reformulated WSD is a degenerated version of our sense-based translation model which only uses the lexicon features,

s_1	s_2	s_3
运营 (operate)	运营 (operate)	运营 (operate)
设施 (facility)	卫星 (satellite)	市场 (market)
计划 (plan)	系统 (system)	企业 (enterprise)
基础 (foundation)	国家 (country)	竞争 (competition)
项目 (project)	提供 (supply)	资产 (assets)
公司 (company)	国际 (inter-nation)	利润 (profit)
结构 (structure)	机构 (institution)	造成 (cause)
服务 (service)	进行 (proceed)	费用 (cost)
组织 (organization)	中心 (center)	资金 (capital)
提供 (supply)	合作 (cooperate)	业务 (business)
s_4	s_5	s_6
费用 (cost)	城市 (city)	处于 (lie)
股价 (share price)	处理 (process)	拍照 (photograph)
27000	自来水 (tap-water)	119
科索沃 (Kosovo)	工厂 (factory)	DPRK
额外 (extra)	汽车 (car)	保险 (insurance)
工资 (wage)	铁路 (railway)	超支 (overspend)
美元 (dollar)	污水 (sewage)	地位 (position)
商业 (commerce)	办事处 (office)	经济 (economy)
收入 (income)	保本 (break-even)	竞争者 (competitor)
铁路局 (railway administration)	部件 (component)	平衡 (balance)

Table 3: Six different senses learned for the word "运营" from the training data.

the sense features used in our model do provide new information that can not be obtained by the lexicon features.

6 Related Work

In this section we introduce previous studies that are related to our work. For ease of comparison, we roughly divide them into 4 categories: 1) WSD for SMT, 2) topic-based WSI, 3) topic model for SMT and 4) lexical selection.

WSD for SMT As we mentioned in Section 1, WSD has been successfully reformulated and adapted to SMT (Vickrey et al., 2005; Carpuat and Wu, 2007; Chan et al., 2007). Rather than predicting word senses for ambiguous words, the reformulated WSD directly predicts target translations for source words with context information. Our sense-based translation model also predicts target translations for SMT. The significant difference is that we predict word senses automatically learned from data and incorporate these predicted senses into SMT. Our experiments show that such word senses are able to improve translation quality.

Topic-based WSI Topic-based WSI can be considered as the foundation of our work as we use it to obtain broad-coverage word senses to annotate our large-scale training data. Brody and Lapata (2009)'s work is the first attempt to approach WSI via topic modeling. They adapt LDA to word sense induction by building one topic model per word type. According to them, there are 3 significant differences between topic-based WSI and generic topic modeling.

- First, the goal of topic-based WSI is to divide contexts of a word type into different categories, each representing a sense cluster. However generic topic models aim at topic distributions of documents.
- Second, generic topic modeling explores whole documents for topic inference while topic-based WSI uses much smaller units in a document (e.g., surrounding words of a target word) for word sense induction.
- Finally, the number of induced word senses in WSI is usually less than 10 while the number of inferred topics in generic topic modeling is tens or hundreds.

As LDA-based WSI needs to manually specify the number of word senses, Yao and Durme (2011) propose HDP-based WSI that is capable of

determining the number of senses for each word type according to training data. Lau et al. (2012) adopt the HDP-based WSI for novel sense detection and empirically show that the HDP-based WSI is better than the LDA-based WSI. We follow them to set the hyperparameters of HDP for training and incorporate automatically induced word senses into SMT in our work.

Topic model for SMT Generic topic models are also explored for SMT. Zhao and Xing (2007) propose a bilingual topic model and integrate a topic-specific lexicon translation model into SMT. Tam et al. (2007) also explore a bilingual topic model for translation and language model adaptation. Foster and Kunh (2007) introduce a mixture model approach for translation model adaptation. Xiao et al. (2012) propose a topic-based similarity model for rule selection in hierarchical phrase-based translation. Xiong and Zhang (2013) employ a sentence-level topic model to capture coherence for document-level machine translation. The difference between our work and these previous studies on topic model for SMT lies in that we adopt topic-based WSI to obtain word senses rather than generic topics and integrate induced word senses into machine translation.

Lexical selection Our work is also related to lexical selection in SMT where appropriate target lexical items for source words are selected by a statistical model with context information (Bangalore et al., 2007; Mauser et al., 2009). The reformulated WSD discussed before can also be considered as a lexical selection model. The significant difference from these studies is that we perform lexical selection using automatically induced word senses by the HDP on the source side.

7 Conclusion

We have presented a sense-based translation model that integrates word senses into machine translation. We capitalize on the broad-coverage word sense induction system that is built on the nonparametric Bayesian HDP to learn sense clusters for words in the source language. We generate pseudo documents for word tokens in the training/test data for the HDP-based WSI system to infer topics. The most probable topic inferred for a pseudo document is taken as the sense of the corresponding word token. We incorporate these learned word senses as translation evidences into maximum entropy classifiers which form the foundation of the proposed sense-based translation model.

We carried out a series of experiments to validate the effectiveness of the sense-based translation by comparing the model against the baseline and the previous reformulated WSD. Our experiment results show that

- The sense-based translation model is able to substantially improve translation quality in terms of both BLEU and NIST.
- The sense-based translation model is also better than the previous reformulated WSD for SMT.
- Word senses automatically induced by the HDP-based WSI on large-scale training data are very useful for machine translation. To the best of our knowledge, this is the first attempt to empirically verify the positive impact of word senses on translation quality.

Comparing with macro topics of documents inferred by LDA with bag of words from the whole documents, word senses inferred by the HDP-based WSI can be considered as micro topics. In the future, we would like to explore both the micro and macro topics for machine translation. Additionally, we also want to induce sense clusters for words in the target language so that we can build sense-based language model and integrate it into SMT. We would like to investigate whether automatically learned senses of proceeding words are helpful for predicting succeeding words.

Acknowledgement

The work was sponsored by the National Natural Science Foundation of China under projects 61373095 and 61333018. We would like to thank three anonymous reviewers for their insightful comments.

References

Srinivas Bangalore, Patrick Haffner, and Stephan Kanthak. 2007. Statistical Machine Translation through Global Lexical Selection and Sentence Reconstruction. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 152–159, Prague, Czech Republic, June. Association for Computational Linguistics.

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–71.

David M. Blei, Andrew Y. Ng, Michael I. Jordan, and John Lafferty. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

Samuel Brody and Mirella Lapata. 2009. Bayesian Word Sense Induction. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 103–111, Athens, Greece, March. Association for Computational Linguistics.

Marine Carpuat and Dekai Wu. 2005. Word Sense Disambiguation vs. Statistical Machine Translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 387–394, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Marine Carpuat and Dekai Wu. 2007. Improving Statistical Machine Translation Using Word Sense Disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 61–72.

Yee Seng Chan, Hwee Tou Ng, and David Chiang. 2007. Word Sense Disambiguation Improves Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 33–40, Prague, Czech Republic, June. Association for Computational Linguistics.

Stanley F. Chen and Joshua Goodman. 1996. An Empirical Study of Smoothing Techniques for Language Modeling. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, ACL '96, pages 310–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

David Chiang. 2005. A Hierarchical Phrase-Based Model for Statistical Machine Translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 263–270, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better Hypothesis Testing for Statistical Machine Translation: Controlling for Optimizer Instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 176–181, Portland, Oregon, USA, June.

George Doddington. 2002. Automatic Evaluation of Machine Translation Quality Using N-gram Co-occurrence Statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*, HLT '02, pages 138–145, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

George Foster and Roland Kuhn. 2007. Mixture-Model Adaptation for SMT. In *Proc. of the Second Workshop on Statistical Machine Translation*, pages 128–135, Prague, Czech Republic, June.

Philipp Koehn, Franz Joseph Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 58–54, Edmonton, Canada, May-June.

Jey Han Lau, Paul Cook, Diana McCarthy, David Newman, and Timothy Baldwin. 2012. Word Sense Induction for Novel Sense Detection. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 591–601, Avignon, France, April. Association for Computational Linguistics.

Arne Mauser, Saša Hasan, and Hermann Ney. 2009. Extending Statistical Machine Translation with Discriminative and Trigger-Based Lexicon Models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 210–218, Singapore, August. Association for Computational Linguistics.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July.

Yik-Cheung Tam, Ian R. Lane, and Tanja Schultz. 2007. Bilingual LSA-based adaptation for statistical machine translation. *Machine Translation*, 21(4):187–207.

Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. 2004. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101.

David Vickrey, Luke Biewald, Marc Teyssier, and Daphne Koller. 2005. Word-Sense Disambiguation for Machine Translation. In *HLT/EMNLP*. The Association for Computational Linguistics.

C. Wang and D. M. Blei. 2012. A Split-Merge MCMC Algorithm for the Hierarchical Dirichlet Process. *ArXiv e-prints*, January.

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

Xinyan Xiao, Deyi Xiong, Min Zhang, Qun Liu, and Shouxun Lin. 2012. A Topic Similarity Model for Hierarchical Phrase-based Translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 750–758, Jeju Island, Korea, July. Association for Computational Linguistics.

Deyi Xiong and Min Zhang. 2013. A Topic-Based Coherence Model for Statistical Machine Translation. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence (AAAI-13)*, Bellevue, Washington, USA, July.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum Entropy Based Phrase Reordering Model for Statistical Machine Translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 521–528, Sydney, Australia, July.

Xuchen Yao and Benjamin Van Durme. 2011. Nonparametric Bayesian Word Sense Induction. In *Proceedings of TextGraphs-6: Graph-based Methods for Natural Language Processing*, pages 10–14, Portland, Oregon, June. Association for Computational Linguistics.

Bin Zhao and Eric P. Xing. 2007. HM-BiTAM: Bilingual Topic Exploration, Word Alignment, and Translation. In *Proc. NIPS 2007*.

Recurrent Neural Networks for Word Alignment Model

Akihiro Tamura[*], **Taro Watanabe, Eiichiro Sumita**
National Institute of Information and Communications Technology
3-5 Hikaridai, Seika-cho, Soraku-gun, Kyoto, JAPAN
a-tamura@ah.jp.nec.com,
{taro.watanabe, eiichiro.sumita}@nict.go.jp

Abstract

This study proposes a word alignment model based on a recurrent neural network (RNN), in which an unlimited alignment history is represented by recurrently connected hidden layers. We perform unsupervised learning using noise-contrastive estimation (Gutmann and Hyvärinen, 2010; Mnih and Teh, 2012), which utilizes artificially generated negative samples. Our alignment model is directional, similar to the generative IBM models (Brown et al., 1993). To overcome this limitation, we encourage agreement between the two directional models by introducing a penalty function that ensures word embedding consistency across two directional models during training. The RNN-based model outperforms the feed-forward neural network-based model (Yang et al., 2013) as well as the IBM Model 4 under Japanese-English and French-English word alignment tasks, and achieves comparable translation performance to those baselines for Japanese-English and Chinese-English translation tasks.

1 Introduction

Automatic word alignment is an important task for statistical machine translation. The most classical approaches are the probabilistic IBM models 1-5 (Brown et al., 1993) and the HMM model (Vogel et al., 1996). Various studies have extended those models. Yang et al. (2013) adapted the Context-Dependent Deep Neural Network for HMM (CD-DNN-HMM) (Dahl et al., 2012), a type of feed-forward neural network (FFNN)-based model, to the HMM alignment model and achieved state-of-the-art performance. However, the FFNN-based model assumes a first-order Markov dependence for alignments.

Recurrent neural network (RNN)-based models have recently demonstrated state-of-the-art performance that outperformed FFNN-based models for various tasks (Mikolov et al., 2010; Mikolov and Zweig, 2012; Auli et al., 2013; Kalchbrenner and Blunsom, 2013; Sundermeyer et al., 2013). An RNN has a hidden layer with recurrent connections that propagates its own previous signals. Through the recurrent architecture, RNN-based models have the inherent property of modeling long-span dependencies, e.g., long contexts, in input data. We assume that this property would fit with a word alignment task, and we propose an RNN-based word alignment model. Our model can maintain and arbitrarily integrate an alignment history, e.g., bilingual context, which is longer than the FFNN-based model.

The NN-based alignment models are supervised models. Unfortunately, it is usually difficult to prepare word-by-word aligned bilingual data. Yang et al. (2013) trained their model from word alignments produced by traditional unsupervised probabilistic models. However, with this approach, errors induced by probabilistic models are learned as correct alignments; thus, generalization capabilities are limited. To solve this problem, we apply noise-contrastive estimation (NCE) (Gutmann and Hyvärinen, 2010; Mnih and Teh, 2012) for unsupervised training of our RNN-based model without gold standard alignments or pseudo-oracle alignments. NCE artificially generates bilingual sentences through samplings as pseudo-negative samples, and then trains the model such that the scores of the original bilingual sentences are higher than those of the sampled bilingual sentences.

Our RNN-based alignment model has a direc-

[*] The first author is now affiliated with Knowledge Discovery Research Laboratories, NEC Corporation, Nara, Japan.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1470–1480,
Baltimore, Maryland, USA, June 23-25 2014.

tion, such as other alignment models, i.e., from f (source language) to e (target language) and from e to f. It has been proven that the limitation may be overcome by encouraging two directional models to agree by training them concurrently (Matusov et al., 2004; Liang et al., 2006; Graça et al., 2008; Ganchev et al., 2008). The motivation for this stems from the fact that model and generalization errors by the two models differ, and the models must complement each other. Based on this motivation, our directional models are also simultaneously trained. Specifically, our training encourages word embeddings to be consistent across alignment directions by introducing a penalty term that expresses the difference between embedding of words into an objective function. This constraint prevents each model from overfitting to a particular direction and leads to global optimization across alignment directions.

This paper presents evaluations of Japanese-English and French-English word alignment tasks and Japanese-to-English and Chinese-to-English translation tasks. The results illustrate that our RNN-based model outperforms the FFNN-based model (up to +0.0792 F1-measure) and the IBM Model 4 (up to +0.0703 F1-measure) for the word alignment tasks. For the translation tasks, our model achieves up to 0.74% gain in BLEU as compared to the FFNN-based model, which matches the translation qualities of the IBM Model 4.

2 Related Work

Various word alignment models have been proposed. These models are roughly clustered into two groups: generative models, such as those proposed by Brown et al. (1993), Vogel et al. (1996), and Och and Ney (2003), and discriminative models, such as those proposed by Taskar et al. (2005), Moore (2005), and Blunsom and Cohn (2006).

2.1 Generative Alignment Model

Given a source language sentence $f_1^J = f_1, ..., f_J$ and a target language sentence $e_1^I = e_1, ..., e_I$, f_1^J is generated by e_1^I via the alignment $a_1^J = a_1, ..., a_J$. Each a_j is a hidden variable indicating that the source word f_j is aligned to the target word e_{a_j}. Usually, a "null" word e_0 is added to the target language sentence and a_1^J may contain $a_j = 0$, which indicates that f_j is not aligned to any target word. The probability of generating the sentence f_1^J from e_1^I is defined as

$$p(f_1^J|e_1^I) = \sum_{a_1^J} p(f_1^J, a_1^J|e_1^I). \quad (1)$$

The IBM Models 1 and 2 and the HMM model decompose it into an alignment probability p_a and a lexical translation probability p_t as

$$p(f_1^J, a_1^J|e_1^I) = \prod_{j=1}^{J} p_a(a_j|a_{j-1}, j)p_t(f_j|e_{a_j}). \quad (2)$$

The three models differ in their definition of alignment probability. For example, the HMM model uses an alignment probability with a first-order Markov property: $p_a(a_j|a_j - a_{j-1})$. In addition, the IBM models 3-5 are extensions of these, which consider the fertility and distortion of each translated word.

These models are trained using the expectation-maximization algorithm (Dempster et al., 1977) from bilingual sentences without word-level alignments (unlabeled training data). Given a specific model, the best alignment (Viterbi alignment) of the sentence pair (f_1^J, e_1^I) can be found as

$$\hat{a}_1^J = \underset{a_1^J}{\operatorname{argmax}}\, p(f_1^J, a_1^J|e_1^I). \quad (3)$$

For example, the HMM model identifies the Viterbi alignment using the Viterbi algorithm.

2.2 FFNN-based Alignment Model

As an instance of discriminative models, we describe an FFNN-based word alignment model (Yang et al., 2013), which is our baseline. An FFNN learns a hierarchy of nonlinear features that can automatically capture complex statistical patterns in input data. Recently, FFNNs have been applied successfully to several tasks, such as speech recognition (Dahl et al., 2012), statistical machine translation (Le et al., 2012; Vaswani et al., 2013), and other popular natural language processing tasks (Collobert and Weston, 2008; Collobert et al., 2011).

Yang et al. (2013) have adapted a type of FFNN, i.e., CD-DNN-HMM (Dahl et al., 2012), to the HMM alignment model. Specifically, the lexical translation and alignment probability in Eq. 2 are computed using FFNNs as

$$s_{NN}(a_1^J|f_1^J, e_1^I) = \prod_{j=1}^{J} t_a(a_j - a_{j-1}|c(e_{a_{j-1}})) \cdot t_{lex}(f_j, e_{a_j}|c(f_j), c(e_{a_j})), \quad (4)$$

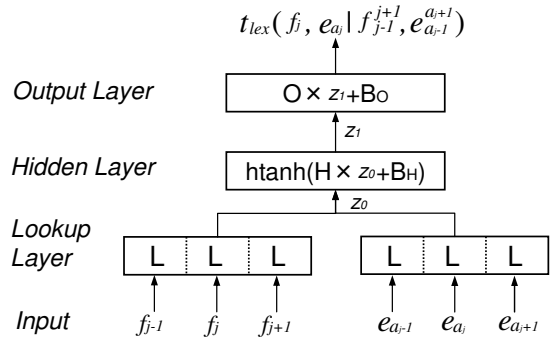


Figure 1: FFNN-based model for computing a lexical translation score of (f_j, e_{a_j})

where t_a and t_{lex} are an alignment score and a lexical translation score, respectively, s_{NN} is a score of alignments a_1^J, and "c(a word w)" denotes a context of word w. Note that the model uses nonprobabilistic scores rather than probabilities because normalization over all words is computationally expensive. The model finds the Viterbi alignment using the Viterbi algorithm, similar to the classic HMM model. Note that alignments in the FFNN-based model are also governed by first-order Markov dynamics because an alignment score depends on the previous alignment a_{j-1}.

Figure 1 shows the network structure with one hidden layer for computing a lexical translation probability $t_{lex}(f_j, e_{a_j}|c(f_j), c(e_{a_j}))$. The model consists of a lookup layer, a hidden layer, and an output layer, which have weight matrices. The model receives a source and target word with their contexts as inputs, which are words in a predefined window (the window size is three in Figure 1). First, the lookup layer converts each input word into its word embedding by looking up its corresponding column in the embedding matrix (L), and then concatenates them. Let V_f (or V_e) be a set of source words (or target words) and M be a predetermined embedding length. L is a $M \times (|V_f| + |V_e|)$ matrix[1]. Word embeddings are dense, low dimensional, and real-valued vectors that can capture syntactic and semantic properties of the words (Bengio et al., 2003). The concatenation (z_0) is then fed to the hidden layer to capture nonlinear relations. Finally, the output layer receives the output of the hidden layer (z_1) and computes a lexical translation score.

[1]We add a special token $\langle unk \rangle$ to handle unknown words and $\langle null \rangle$ to handle null alignments to V_f and V_e

The computations in the hidden and output layer are as follows[2]:

$$z_1 = f(H \times z_0 + B_H), \quad (5)$$

$$t_{lex} = O \times z_1 + B_O, \quad (6)$$

where H, B_H, O, and B_O are $|z_1| \times |z_0|$, $|z_1| \times 1$, $1 \times |z_1|$, and 1×1 matrices, respectively, and $f(x)$ is an activation function. Following Yang et al. (2013), a "hard" version of the hyperbolic tangent, htanh(x)[3], is used as $f(x)$ in this study.

The alignment model based on an FFNN is formed in the same manner as the lexical translation model. Each model is optimized by minimizing the following ranking loss with a margin using stochastic gradient descent (SGD)[4], where gradients are computed by the back-propagation algorithm (Rumelhart et al., 1986):

$$loss(\theta) = \sum_{(\boldsymbol{f},\boldsymbol{e}) \in T} \max\{0, 1 - s_\theta(\boldsymbol{a}^+|\boldsymbol{f}, \boldsymbol{e}) + s_\theta(\boldsymbol{a}^-|\boldsymbol{f}, \boldsymbol{e})\}, \quad (7)$$

where θ denotes the weights of layers in the model, T is a set of training data, $\boldsymbol{a}^+$ is the gold standard alignment, $\boldsymbol{a}^-$ is the incorrect alignment with the highest score under θ, and s_θ denotes the score defined by Eq. 4 as computed by the model under θ.

3 RNN-based Alignment Model

This section proposes an RNN-based alignment model, which computes a score for alignments a_1^J using an RNN:

$$s_{NN}(a_1^J|f_1^J, e_1^I) = \prod_{j=1}^{J} t_{RNN}(a_j|a_1^{j-1}, f_j, e_{a_j}), \quad (8)$$

where t_{RNN} is the score of an alignment a_j. The prediction of the j-th alignment a_j depends on all preceding alignments a_1^{j-1}. Note that the proposed model also uses nonprobabilistic scores, similar to the FFNN-based model.

The RNN-based model is illustrated in Figure 2. The model consists of a lookup layer, a hidden layer, and an output layer, which have weight

[2]Consecutive l hidden layers can be used: $z_l = f(H_l \times z_{l-1} + B_{H_l})$. For simplicity, this paper describes the model with 1 hidden layer.

[3]htanh$(x) = -1$ for $x < -1$, htanh$(x) = 1$ for $x > 1$, and htanh$(x) = x$ for others.

[4]In our experiments, we used a mini-batch SGD instead of a plain SGD.

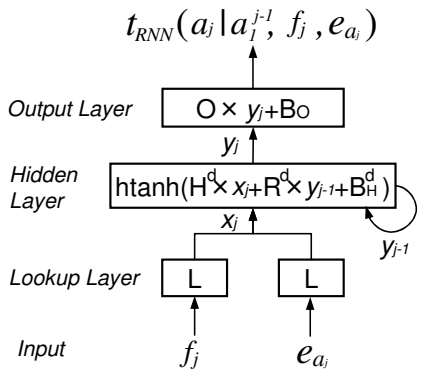


Figure 2: RNN-based alignment model

matrices L, $\{H^d, R^d, B^d_H\}$, and $\{O, B_O\}$, respectively. Each matrix in the hidden layer (H^d, R^d, and B^d_H) depends on alignment, where d denotes the jump distance from a_{j-1} to a_j: $d = a_j - a_{j-1}$. In our experiments, we merge distances that are greater than 8 and less than -8 into the special "$\geq$8" and "$\leq$-8" distances, respectively. Specifically, the hidden layer has weight matrices $\{H^{\leq -8}, H^{-7}, \cdots, H^{7}, H^{\geq 8}, R^{\leq -8}, R^{-7}, \cdots, R^{7}, R^{\geq 8}, B^{\leq -8}_H, B^{-7}_H, \cdots, B^{7}_H, B^{\geq 8}_H\}$ and computes y_j using the corresponding matrices of the jump distance d.

The Viterbi alignment is determined using the Viterbi algorithm, similar to the FFNN-based model, where the model is sequentially applied from f_1 to f_J[5]. When computing the score of the alignment between f_j and e_{a_j}, the two words are input to the lookup layer. In the lookup layer, each of these words is converted to its word embedding, and then the concatenation of the two embeddings (x_j) is fed to the hidden layer in the same manner as the FFNN-based model. Next, the hidden layer receives the output of the lookup layer (x_j) and that of the previous hidden layer (y_{j-1}). The hidden layer then computes and outputs the nonlinear relations between them. Note that the weight matrices used in this computation are embodied by the specific jump distance d. The output of the hidden layer (y_j) is copied and fed to the output layer and the next hidden layer. Finally, the output layer computes the score of a_j ($t_{RNN}(a_j|a_1^{j-1}, f_j, e_{a_j})$) from the output of the hidden layer (y_j). Note that the FFNN-based model consists of two components: one is for lexical translation and the other is for alignment. The proposed RNN produces a single score that is constructed in the hidden layer by employing the distance-dependent weight matrices.

Specifically, the computations in the hidden and output layer are as follows:

$$y_j = f(H^d \times x_j + R^d \times y_{j-1} + B^d_H), \quad (9)$$

$$t_{RNN} = O \times y_j + B_O, \quad (10)$$

where H^d, R^d, B^d_H, O, and B_O are $|y_j| \times |x_j|$, $|y_j| \times |y_{j-1}|$, $|y_j| \times 1$, $1 \times |y_j|$, and 1×1 matrices, respectively. Note that $|y_{j-1}| = |y_j|$. $f(x)$ is an activation function, which is a hard hyperbolic tangent, i.e., htanh(x), in this study.

As described above, the RNN-based model has a hidden layer with recurrent connections. Under the recurrence, the proposed model compactly encodes the entire history of previous alignments in the hidden layer configuration y_i. Therefore, the proposed model can find alignments by taking advantage of the long alignment history, while the FFNN-based model considers only the last alignment.

4 Training

During training, we optimize the weight matrices of each layer (i.e., L, H^d, R^d, B^d_H, O, and B_O) following a given objective using a mini-batch SGD with batch size D, which converges faster than a plain SGD (D = 1). Gradients are computed by the back-propagation through time algorithm (Rumelhart et al., 1986), which unfolds the network in time (j) and computes gradients over time steps. In addition, an $l2$ regularization term is added to the objective to prevent the model from overfitting the training data.

The RNN-based model can be trained by a supervised approach, similar to the FFNN-based model, where training proceeds based on the ranking loss defined by Eq. 7 (Section 2.2). However, this approach requires gold standard alignments. To overcome this drawback, we propose an unsupervised method using NCE, which learns from unlabeled training data.

4.1 Unsupervised Learning

Dyer et al. (2011) presented an unsupervised alignment model based on contrastive estimation (CE) (Smith and Eisner, 2005). CE seeks to discriminate observed data from its neighborhood,

[5]Strictly speaking, we cannot apply the dynamic programming forward-backward algorithm (i.e., the Viterbi algorithm) due to the long alignment history of y_i. Thus, the Viterbi alignment is computed approximately using heuristic beam search.

which can be viewed as pseudo-negative samples. Dyer et al. (2011) regarded all possible alignments of the bilingual sentences, which are given as training data (T), and those of the full translation search space (Ω) as the observed data and its neighborhood, respectively.

We introduce this idea to a ranking loss with margin as

$$loss(\theta) = \max\Bigg\{0, 1 - \sum_{(\boldsymbol{f}^+, \boldsymbol{e}^+) \in T} \mathrm{E}_{\Phi}[s_\theta(\boldsymbol{a}|\boldsymbol{f}^+, \boldsymbol{e}^+)] + \sum_{(\boldsymbol{f}^+, \boldsymbol{e}^-) \in \Omega} \mathrm{E}_{\Phi}[s_\theta(\boldsymbol{a}|\boldsymbol{f}^+, \boldsymbol{e}^-)]\Bigg\}, \quad (11)$$

where Φ is a set of all possible alignments given $(\boldsymbol{f}, \boldsymbol{e})$, $\mathrm{E}_{\Phi}[s_\theta]$ is the expected value of the scores s_θ on Φ, $\boldsymbol{e}^+$ denotes a target language sentence in the training data, and $\boldsymbol{e}^-$ denotes a pseudo-target language sentence. The first expectation term is for the observed data, and the second is for the neighborhood.

However, the computation for Ω is prohibitively expensive. To reduce computation, we employ NCE, which uses randomly sampled sentences from all target language sentences in Ω as $\boldsymbol{e}^-$, and calculate the expected values by a beam search with beam width W to truncate alignments with low scores. In our experiments, we set W to 100. In addition, the above criterion is converted to an online fashion as

$$loss(\theta) = \sum_{\boldsymbol{f}^+ \in T} \max\Bigg\{0, 1 - \mathrm{E}_{\mathrm{GEN}}[s_\theta(\boldsymbol{a}|\boldsymbol{f}^+, \boldsymbol{e}^+)] + \frac{1}{N} \sum_{\boldsymbol{e}^-} \mathrm{E}_{\mathrm{GEN}}[s_\theta(\boldsymbol{a}|\boldsymbol{f}^+, \boldsymbol{e}^-)]\Bigg\}, \quad (12)$$

where $\boldsymbol{e}^+$ is a target language sentence aligned to $\boldsymbol{f}^+$ in the training data, i.e., $(\boldsymbol{f}^+, \boldsymbol{e}^+) \in T$, $\boldsymbol{e}^-$ is a randomly sampled pseudo-target language sentence with length $|\boldsymbol{e}^+|$, and N denotes the number of pseudo-target language sentences per source sentence $\boldsymbol{f}^+$. Note that $|\boldsymbol{e}^+| = |\boldsymbol{e}^-|$. GEN is a subset of all possible word alignments Φ, which is generated by beam search.

In a simple implementation, each $\boldsymbol{e}^-$ is generated by repeating a random sampling from a set of target words (V_e) $|\boldsymbol{e}^+|$ times and lining them up sequentially. To employ more discriminative negative samples, our implementation samples each word of $\boldsymbol{e}^-$ from a set of the target words that co-occur with $f_i \in \boldsymbol{f}^+$ whose probability is above a threshold C under the IBM Model 1 incorporating l_0 prior (Vaswani et al., 2012). The IBM Model 1 with l_0 prior is convenient for reducing translation candidates because it generates more sparse alignments than the standard IBM Model 1.

4.2 Agreement Constraints

Both of the FFNN-based and RNN-based models are based on the HMM alignment model, and they are therefore asymmetric, i.e., they can represent one-to-many relations from the target side. Asymmetric models are usually trained in each alignment direction. The model proposed by Yang et al. (2013) is no exception. However, it has been demonstrated that encouraging directional models to agree improves alignment performance (Matusov et al., 2004; Liang et al., 2006; Graça et al., 2008; Ganchev et al., 2008).

Inspired by their work, we introduce an agreement constraint to our learning. The constraint concretely enforces agreement in word embeddings of both directions. The proposed method trains two directional models concurrently based on the following objective by incorporating a penalty term that expresses the difference between word embeddings:

$$\operatorname*{argmin}_{\theta_{FE}} \big\{loss(\theta_{FE}) + \alpha\|\theta_{L_{EF}} - \theta_{L_{FE}}\|\big\}, \quad (13)$$

$$\operatorname*{argmin}_{\theta_{EF}} \big\{loss(\theta_{EF}) + \alpha\|\theta_{L_{FE}} - \theta_{L_{EF}}\|\big\}, \quad (14)$$

where θ_{FE} (or θ_{EF}) denotes the weights of layers in a source-to-target (or target-to-source) alignment model, θ_L denotes weights of a lookup layer, i.e., word embeddings, and α is a parameter that controls the strength of the agreement constraint. $\|\theta\|$ indicates the norm of θ. 2-norm is used in our experiments. Equations 13 and 14 can be applied to both supervised and unsupervised approaches. Equations 7 and 12 are substituted into $loss(\theta)$ in supervised and unsupervised learning, respectively. The proposed constraint penalizes overfitting to a particular direction and enables two directional models to optimize across alignment directions globally.

Our unsupervised learning procedure is summarized in Algorithm 1. In Algorithm 1, line 2 randomly samples D bilingual sentences $(\mathbf{f}^+, \mathbf{e}^+)^D$ from training data T. Lines 3-1 and 3-2 generate N pseudo-negative samples for each $\mathbf{f}^+$ and $\mathbf{e}^+$ based on the translation candidates of $\mathbf{f}^+$ and $\mathbf{e}^+$ found by the IBM Model 1 with l_0 prior,

Algorithm 1 Training Algorithm

Input: θ^1_{FE}, θ^1_{EF}, training data T, $MaxIter$, batch size D, N, C, $IBM1$, W, α

1: **for all** t such that $1 \leq t \leq MaxIter$ **do**

2: $\{(\mathbf{f}^+, \mathbf{e}^+)^D\} \leftarrow \text{sample}(D, T)$

3-1: $\{(\mathbf{f}^+, \{\mathbf{e}^-\}^N)^D\} \leftarrow \text{neg}_\text{e}(\{(\mathbf{f}^+, \mathbf{e}^+)^D\}, N, C, IBM1)$

3-2: $\{(\mathbf{e}^+, \{\mathbf{f}^-\}^N)^D\} \leftarrow \text{neg}_\text{f}(\{(\mathbf{f}^+, \mathbf{e}^+)^D\}, N, C, IBM1)$

4-1: $\theta^{t+1}_{FE} \leftarrow \text{update}((\mathbf{f}^+, \mathbf{e}^+, \{\mathbf{e}^-\}^N)^D, \theta^t_{FE}, \theta^t_{EF}, W, \alpha)$

4-2: $\theta^{t+1}_{EF} \leftarrow \text{update}((\mathbf{e}^+, \mathbf{f}^+, \{\mathbf{f}^-\}^N)^D, \theta^t_{EF}, \theta^t_{FE}, W, \alpha)$

5: **end for**

Output: $\theta^{MaxIter+1}_{EF}, \theta^{MaxIter+1}_{FE}$

		Train	Dev	Test
$BTEC$		9 K	0	960
$Hansards$		1.1 M	37	447
$FBIS$	$NIST03$	240 K	878	919
	$NIST04$			1,597
$IWSLT$		40 K	2,501	489
$NTCIR$		3.2 M	2,000	2,000

Table 1: Size of experimental datasets

$IBM1$ (Section 4.1). Lines 4-1 and 4-2 update the weights in each layer following a given objective (Sections 4.1 and 4.2). Note that θ^t_{FE} and θ^t_{EF} are concurrently updated in each iteration, and θ^t_{EF} (or θ^t_{FE}) is employed to enforce agreement between word embeddings when updating θ^t_{FE} (or θ^t_{EF}).

5 Experiment

5.1 Experimental Data

We evaluated the alignment performance of the proposed models with two tasks: Japanese-English word alignment with the Basic Travel Expression Corpus ($BTEC$) (Takezawa et al., 2002) and French-English word alignment with the Hansard dataset ($Hansards$) from the 2003 NAACL shared task (Mihalcea and Pedersen, 2003). In addition, we evaluated the end-to-end translation performance of three tasks: a Chinese-to-English translation task with the FBIS corpus ($FBIS$), the IWSLT 2007 Japanese-to-English translation task ($IWSLT$) (Fordyce, 2007), and the NTCIR-9 Japanese-to-English patent translation task ($NTCIR$) (Goto et al., 2011)[6].

Table 1 shows the sizes of our experimental datasets. Note that the development data was not used in the alignment tasks, i.e., $BTEC$ and $Hansards$, because the hyperparameters of the alignment models were set by preliminary small-scale experiments. The $BTEC$ data is the first 9,960 sentence pairs in the training data for $IWSLT$, which were annotated with word alignment (Goh et al., 2010). We split these pairs into the first 9,000 for training data and the remaining 960 as test data. All the data in $BTEC$ is word-aligned, and the training data in $Hansards$ is unlabeled data. In $FBIS$, we used the NIST02 evaluation data as the development data, and the NIST03 and 04 evaluation data as test data ($NIST03$ and $NIST04$).

[6]We did not evaluate the translation performance on the Hansards data because the development data is very small and performance is unreliable.

5.2 Comparing Methods

We evaluated the proposed RNN-based alignment models against two baselines: the IBM Model 4 and the FFNN-based model with one hidden layer. The IBM Model 4 was trained by previously presented model sequence schemes (Och and Ney, 2003): $1^5H^53^54^5$, i.e., five iterations of the IBM Model 1 followed by five iterations of the HMM Model, etc., which is the default setting for GIZA++ ($IBM4$). For the FFNN-based model, we set the word embedding length M to 30, the number of units of a hidden layer $|z_1|$ to 100, and the window size of contexts to 5. Hence, $|z_0|$ is 300 ($30 \times 5 \times 2$). Following Yang et al. (2013), the FFNN-based model was trained by the supervised approach described in Section 2.2 ($FFNN_s$).

For the RNN-based models, we set M to 30 and the number of units of each recurrent hidden layer $|y_j|$ to 100. Thus, $|x_j|$ is 60 (30×2). The number of units of each layer of the FFNN-based and RNN-based models and M were set through preliminary experiments. To demonstrate the effectiveness of the proposed learning methods, we evaluated four types of RNN-based models: RNN_s, RNN_{s+c}, RNN_u, and RNN_{u+c}, where "*s*/*u*" denotes a supervised/unsupervised model and "$+c$" indicates that the agreement constraint was used.

In training all the models except $IBM4$, the weights of each layer were initialized first. For the weights of a lookup layer L, we preliminarily trained word embeddings for the source and target language from each side of the training data. We then set the word embeddings to L to avoid falling into local minima. Other weights were randomly initialized to $[-0.1, 0.1]$. For the pretraining, we

Alignment	$BTEC$	$Hansards$
$IBM4$	0.4859	0.9029
$FFNN_s$(I)	0.4770	0.9020
RNN_s(I)	0.5053$^+$	0.9068
RNN_{s+c}(I)	0.5174$^+$	0.9202$^+$
RNN_u	0.5307$^+$	0.9037
RNN_{u+c}	**0.5562**$^+$	**0.9275**$^+$
$FFNN_s$(R)	0.8224	-
RNN_s(R)	0.8798$^+$	-
RNN_{s+c}(R)	**0.8921**$^+$	-

Table 2: Word alignment performance (F1-measure)

used the RNNLM Toolkit [7] (Mikolov et al., 2010) with the default options. We mapped all words that occurred less than five times to the special token $\langle unk \rangle$. Next, each weight was optimized using the mini-batch SGD, where batch size D was 100, learning rate was 0.01, and an l_2 regularization parameter was 0.1. The training stopped after 50 epochs. The other parameters were set as follows: W, N and C in the unsupervised learning were 100, 50, and 0.001, respectively, and α for the agreement constraint was 0.1.

In the translation tasks, we used the Moses phrase-based SMT systems (Koehn et al., 2007). All Japanese and Chinese sentences were segmented by ChaSen[8] and the Stanford Chinese segmenter[9], respectively. In the training, long sentences with over 40 words were filtered out. Using the SRILM Toolkits (Stolcke, 2002) with modified Kneser-Ney smoothing, we trained a 5-gram language model on the English side of each training data for $IWSLT$ and $NTCIR$, and a 5-gram language model on the Xinhua portion of the English Gigaword corpus for $FBIS$. The SMT weighting parameters were tuned by MERT (Och, 2003) in the development data.

5.3 Word Alignment Results

Table 2 shows the alignment performance by the F1-measure. Hereafter, $MODEL(R)$ and $MODEL(I)$ denote the $MODEL$ trained from gold standard alignments and word alignments found by the IBM Model 4, respectively. In $Hansards$, all models were trained from randomly sampled 100 K data[10]. We evaluated the word alignments produced by first applying each model in both directions and then combining the alignments using the "grow-diag-final-and" heuristic (Koehn et al., 2003). The significance test on word alignment performance was performed by the sign test with a 5% significance level. "+" in Table 2 indicates that the comparisons are significant over corresponding baselines, $IBM4$ and $FFNN_s(R/I)$.

In Table 2, RNN_{u+c}, which includes all our proposals, i.e., the RNN-based model, the unsupervised learning, and the agreement constraint, achieves the best performance for both $BTEC$ and $Hansards$. The differences from the baselines are statistically significant.

Table 2 shows that $RNN_s(R/I)$ outperforms $FFNN_s(R/I)$, which is statistically significant in $BTEC$. These results demonstrate that capturing the long alignment history in the RNN-based model improves the alignment performance. We discuss the difference of the RNN-based model's effectiveness between language pairs in Section 6.1. Table 2 also shows that $RNN_{s+c}(R/I)$ and RNN_{u+c} achieve significantly better performance than $RNN_s(R/I)$ and RNN_u in both tasks, respectively. This indicates that the proposed agreement constraint is effective in training better models in both the supervised and unsupervised approaches.

In $BTEC$, RNN_u and RNN_{u+c} significantly outperform $RNN_s(I)$ and $RNN_{s+c}(I)$, respectively. The performance of these models is comparable with $Hansards$. This indicates that our unsupervised learning benefits our models because the supervised models are adversely affected by errors in the automatically generated training data. This is especially true when the quality of training data, i.e., the performance of $IBM4$, is low.

5.4 Machine Translation Results

Table 3 shows the translation performance by the case sensitive BLEU4 metric[11] (Papineni et al., 2002). Table 3 presents the average BLEU of three different MERT runs. In $NTCIR$ and $FBIS$, each alignment model was trained from the ran-

[7] http://www.fit.vutbr.cz/~imikolov/rnnlm/

[8] http://chasen-legacy.sourceforge.jp/

[9] http://nlp.stanford.edu/software/segmenter.shtml

[10] Due to high computational cost, we did not use all the training data. Scaling up to larger datasets will be addressed in future work.

[11] We used mteval-v13a.pl as the evaluation tool (http://www.itl.nist.gov/iad/mig/tests/mt/2009/).

Alignment	$IWSLT$	$NTCIR$	$FBIS$	
			$NIST03$	$NIST04$
$IBM4_{all}$	46.47	27.91	25.90	28.34
$IBM4$		27.25	25.41	27.65
$FFNN_s$(I)	46.38	27.05	25.45	27.61
RNN_s(I)	46.43	27.24	25.47	27.56
RNN_{s+c}(I)	46.51	27.12	25.55	27.73
RNN_u	47.05*	27.79*	25.76*	27.91*
RNN_{u+c}	46.97*	27.76*	25.84*	28.20*

Table 3: Translation performance (BLEU4(%))

domly sampled 100 K data, and then a translation model was trained from all the training data that was word-aligned by the alignment model. In addition, for a detailed comparison, we evaluated the SMT system where the IBM Model 4 was trained from all the training data ($IBM4_{all}$). The significance test on translation performance was performed by the bootstrap method (Koehn, 2004) with a 5% significance level. "*" in Table 3 indicates that the comparisons are significant over both baselines, i.e., $IBM4$ and $FFNN_s(I)$.

Table 3 also shows that better word alignment does not always result in better translation, which has been discussed previously (Yang et al., 2013). However, RNN_u and RNN_{u+c} outperform $FFNN_s(I)$ and $IBM4$ in all tasks. These results indicate that our proposals contribute to improving translation performance[12]. In addition, Table 3 shows that these proposed models are comparable to $IBM4_{all}$ in $NTCIR$ and $FBIS$ even though the proposed models are trained from only a small part of the training data.

6 Discussion

6.1 Effectiveness of RNN-based Alignment Model

Figure 3 shows word alignment examples from $FFNN_s$ and RNN_s, where solid squares indicate the gold standard alignments. Figure 3 (a) shows that RRN_s adequately identifies complicated alignments with long distances compared to $FFNN_s$ (e.g., jaggy alignments of "have you been learning" in Fig 3 (a)) because RNN_s captures alignment paths based on long alignment history, which can be viewed as phrase-level alignments, while $FFNN_s$ employs only the last alignment.

In French-English word alignment, the most

[12]We also confirmed the effectiveness of our models on the NIST05 and NTCIR-10 evaluation data.

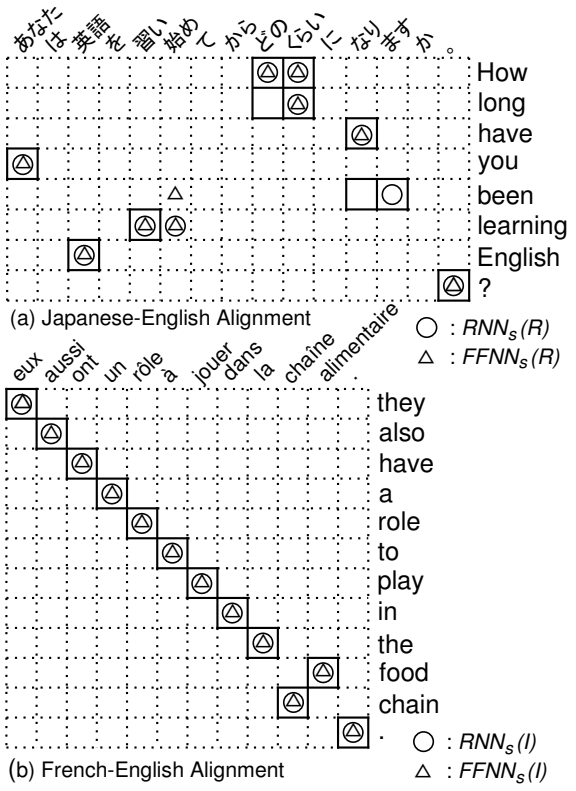


Figure 3: Word alignment examples

Alignment	40 K	9 K	1 K
$IBM4$	0.5467	0.4859	0.4128
RNN_{u+c}	0.6004	0.5562	0.4842
$RNN_{s+c}(R)$	-	0.8921	0.6063

Table 4: Word alignment performance on $BTEC$ with various sized training data

valuable clues are located locally because English and French have similar word orders and their alignment has more one-to-one mappings than Japanese-English word alignment (Figure 3). Figure 3 (b) shows that both RRN_s and $FFNN_s$ work for such simpler alignments. Therefore, the RNN-based model has less effect on French-English word alignment than Japanese-English word alignment, as indicated in Table 2.

6.2 Impact of Training Data Size

Table 4 shows the alignment performance on $BTEC$ with various training data sizes, i.e., training data for $IWSLT$ (40 K), training data for $BTEC$ (9 K), and the randomly sampled 1 K data from the $BTEC$ training data. Note that $RNN_{s+c}(R)$ cannot be trained from the 40 K data because the 40 K data does not have gold standard

Alignment	$BTEC$	$Hansards$
$FFNN_s$(I)	0.4770	0.9020
$FFNN_{s+c}$(I)	0.4854^+	0.9085^+
$FFNN_u$	0.5105^+	0.9026
$FFNN_{u+c}$	0.5313^+	0.9144^+
$FFNN_s$(R)	0.8224	-
$FFNN_{s+c}$(R)	0.8367^+	-

Table 5: Word alignment performance of various FFNN-based models (F1-measure)

word alignments.

Table 4 demonstrates that the proposed RNN-based model outperforms $IBM4$ trained from the unlabeled 40 K data by employing either the 1 K labeled data or the 9 K unlabeled data, which is less than 25% of the training data for $IBM4$. Consequently, the SMT system using RNN_{u+c} trained from a small part of training data can achieve comparable performance to that using $IBM4$ trained from all training data, which is shown in Table 3.

6.3 Effectiveness of Unsupervised Learning/Agreement Constraints

The proposed unsupervised learning and agreement constraints can be applied to any NN-based alignment model. Table 5 shows the alignment performance of the FFNN-based models trained by our supervised/unsupervised approaches (s/u) with and without our agreement constraints. In Table 5, "+c" denotes that the agreement constraint was used, and "+" indicates that the comparison with its corresponding baseline, i.e., $FFNN_s$(I/R), is significant in the sign test with a 5% significance level.

Table 5 shows that $FFNN_{s+c}$(R/I) and $FFNN_{u+c}$ achieve significantly better performance than $FFNN_s$(R/I) and $FFNN_u$, respectively, in both $BTEC$ and $Hansards$. In addition, $FFNN_u$ and $FFNN_{u+c}$ significantly outperform $FFNN_s$(I) and $FFNN_{s+c}$(I), respectively, in $BTEC$. The performance of these models is comparable in $Hansards$. These results indicate that the proposed unsupervised learning and agreement constraint benefit the FFNN-based model, similar to the RNN-based model.

7 Conclusion

We have proposed a word alignment model based on an RNN, which captures long alignment history through recurrent architectures. Furthermore, we proposed an unsupervised method for training our model using NCE and introduced an agreement constraint that encourages word embeddings to be consistent across alignment directions. Our experiments have shown that the proposed model outperforms the FFNN-based model (Yang et al., 2013) for word alignment and machine translation, and that the agreement constraint improves alignment performance.

In future, we plan to employ contexts composed of surrounding words (e.g., $c(f_j)$ or $c(e_{a_j})$ in the FFNN-based model) in our model, even though our model implicitly encodes such contexts in the alignment history. We also plan to enrich each hidden layer in our model with multiple layers following the success of Yang et al. (2013), in which multiple hidden layers improved the performance of the FFNN-based model. In addition, we would like to prove the effectiveness of the proposed method for other datasets.

Acknowledgments

We thank the anonymous reviewers for their helpful suggestions and valuable comments on the first version of this paper.

References

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

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3:1137–1155.

Phil Blunsom and Trevor Cohn. 2006. Discriminative Word Alignment with Conditional Random Fields. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 65–72.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311.

Ronan Collobert and Jason Weston. 2008. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In

Proceedings of the 25th International Conference on Machine Learning, pages 160–167.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537.

George E. Dahl, Dong Yu, Li Deng, and Alex Acero. 2012. Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1):30–42.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

Chris Dyer, Jonathan Clark, Alon Lavie, and Noah A. Smith. 2011. Unsupervised Word Alignment with Arbitrary Features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pages 409–419.

Cameron S. Fordyce. 2007. Overview of the IWSLT 2007 Evaluation Campaign. In *Proceedings of the 4th International Workshop on Spoken Languaeg Translation*, pages 1–12.

Kuzman Ganchev, João V. Graça, and Ben Taskar. 2008. Better Alignments = Better Translations? In *Proceedings of the 46th Annual Conference of the Association for Computational Linguistics: Human Language Technologies*, pages 986–993.

Chooi-Ling Goh, Taro Watanabe, Hirofumi Yamamoto, and Eiichiro Sumita. 2010. Constraining a Generative Word Alignment Model with Discriminative Output. *IEICE Transactions*, 93-D(7):1976–1983.

Isao Goto, Bin Lu, Ka Po Chow, Eiichiro Sumita, and Benjamin K. Tsou. 2011. Overview of the Patent Machine Translation Task at the NTCIR-9 Workshop. In *Proceedings of the 9th NTCIR Workshop*, pages 559–578.

João V. Graça, Kuzman Ganchev, and Ben Taskar. 2008. Expectation Maximization and Posterior Constraints. In *Advances in Neural Information Processing Systems 20*, pages 569–576.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-Contrastive Estimation: A New Estimation Principle for Unnormalized Statistical Models. In *Proceedings of the 13st International Conference on Artificial Intelligence and Statistics*, pages 297–304.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent Continuous Translation Models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of the 2003 Human Language Technology Conference: North American Chapter of the Association for Computational Linguistics*, pages 48–54.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constrantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics on Interactive Poster and Demonstration Sessions*, pages 177–180.

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395.

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

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by Agreement. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 104–111.

Evgeny Matusov, Richard Zens, and Hermann Ney. 2004. Symmetric Word Alignments for Statistical Machine Translation. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 219–225.

Rada Mihalcea and Ted Pedersen. 2003. An Evaluation Exercise for Word Alignment. In *Proceedings of the HLT-NAACL 2003 Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, pages 1–10.

Tomas Mikolov and Geoffrey Zweig. 2012. Context Dependent Recurrent Neural Network Language Model. In *Proceedings of the 4th IEEE Workshop on Spoken Language Technology*, pages 234–239.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent Neural Network based Language Model. In *Proceedings of 11th Annual Conference of the International Speech Communication Association*, pages 1045–1048.

Andriy Mnih and Yee Whye Teh. 2012. A Fast and Simple Algorithm for Training Neural Probabilistic Language Models. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1751–1758.

Robert C. Moore. 2005. A Discriminative Framework for Bilingual Word Alignment. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 81–88.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29:19–51.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. 1986. Learning Internal Representations by Error Propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, pages 318–362. MIT Press.

Noah A. Smith and Jason Eisner. 2005. Contrastive Estimation: Training Log-Linear Models on Unlabeled Data. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 354–362.

Andreas Stolcke. 2002. SRILM - An Extensible Language Modeling Toolkit. In *Proceedings of International Conference on Spoken Language Processing*, pages 901–904.

Martin Sundermeyer, Ilya Oparin, Jean-Luc Gauvain, Ben Freiberg, Ralf Schlüter, and Hermann Ney. 2013. Comparison of Feedforward and Recurrent Neural Network Language Models. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 8430–8434.

Toshiyuki Takezawa, Eiichiro Sumita, Fumiaki Sugaya, Hirofumi Yamamoto, and Seiichi Yamamoto. 2002. Toward a Broad-coverage Bilingual Corpus for Speech Translation of Travel Conversations in the Real World. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation*, pages 147–152.

Ben Taskar, Simon Lacoste-Julien, and Dan Klein. 2005. A Discriminative Matching Approach to Word Alignment. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 73–80.

Ashish Vaswani, Liang Huang, and David Chiang. 2012. Smaller Alignment Models for Better Translations: Unsupervised Word Alignment with the l_0-norm. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 311–319.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with Large-Scale Neural Language Models Improves Translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1387–1392.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. Hmm-based Word Alignment in Statistical Translation. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 836–841.

Nan Yang, Shujie Liu, Mu Li, Ming Zhou, and Nenghai Yu. 2013. Word Alignment Modeling with Context Dependent Deep Neural Network. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 166–175.

A Constrained Viterbi Relaxation for Bidirectional Word Alignment

Yin-Wen Chang **Alexander M. Rush**
MIT CSAIL,
Cambridge, MA 02139
{yinwen,srush}@
csail.mit.edu

John DeNero
UC Berkeley,
Berkeley, CA 94720
denero@
cs.berkeley.edu

Michael Collins
Columbia University,
New York, NY 10027
mcollins@
cs.columbia.edu

Abstract

Bidirectional models of word alignment are an appealing alternative to post-hoc combinations of directional word aligners. Unfortunately, most bidirectional formulations are NP-Hard to solve, and a previous attempt to use a relaxation-based decoder yielded few exact solutions (6%). We present a novel relaxation for decoding the bidirectional model of DeNero and Macherey (2011). The relaxation can be solved with a modified version of the Viterbi algorithm. To find optimal solutions on difficult instances, we alternate between incrementally adding constraints and applying optimality-preserving coarse-to-fine pruning. The algorithm finds provably exact solutions on 86% of sentence pairs and shows improvements over directional models.

1 Introduction

Word alignment is a critical first step for building statistical machine translation systems. In order to ensure accurate word alignments, most systems employ a post-hoc symmetrization step to combine directional word aligners, such as IBM Model 4 (Brown et al., 1993) or hidden Markov model (HMM) based aligners (Vogel et al., 1996). Several authors have proposed bidirectional models that incorporate this step directly, but decoding under many bidirectional models is NP-Hard and finding exact solutions has proven difficult.

In this paper, we describe a novel Lagrangian-relaxation based decoder for the bidirectional model proposed by DeNero and Macherey (2011), with the goal of improving search accuracy. In that work, the authors implement a dual decomposition-based decoder for the problem, but are only able to find exact solutions for around 6% of instances.

Our decoder uses a simple variant of the Viterbi algorithm for solving a relaxed version of this model. The algorithm makes it easy to re-introduce constraints for difficult instances, at the cost of increasing run-time complexity. To offset this cost, we employ optimality-preserving coarse-to-fine pruning to reduce the search space. The pruning method utilizes lower bounds on the cost of valid bidirectional alignments, which we obtain from a fast, greedy decoder.

The method has the following properties:

- It is based on a novel relaxation for the model of DeNero and Macherey (2011), solvable with a variant of the Viterbi algorithm.
- To find optimal solutions, it employs an efficient strategy that alternates between adding constraints and applying pruning.
- Empirically, it is able to find exact solutions on 86% of sentence pairs and is significantly faster than general-purpose solvers.

We begin in Section 2 by formally describing the directional word alignment problem. Section 3 describes a preliminary bidirectional model using full agreement constraints and a Lagrangian relaxation-based solver. Section 4 modifies this model to include adjacency constraints. Section 5 describes an extension to the relaxed algorithm to explicitly enforce constraints, and Section 6 gives a pruning method for improving the efficiency of the algorithm.

Experiments compare the search error and accuracy of the new bidirectional algorithm to several directional combiners and other bidirectional algorithms. Results show that the new relaxation is much more effective at finding exact solutions and is able to produce comparable alignment accuracy.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1481–1490,
Baltimore, Maryland, USA, June 23-25 2014.

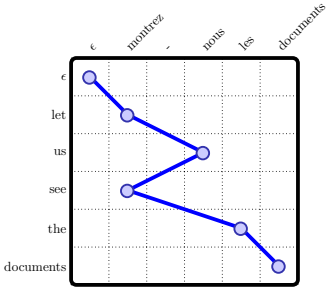


Figure 1: An example $\mathbf{e}\rightarrow\mathbf{f}$ directional alignment for the sentences `let us see the documents` and `montrez - nous les documents`, with $I = 5$ and $J = 5$. The indices $i \in [I]_0$ are rows, and the indices $j \in [J]_0$ are columns. The HMM alignment shown has transitions $x(0,1,1) = x(1,2,3) = x(3,3,1) = x(1,4,4) = x(4,5,5) = 1$.

Notation We use lower- and upper-case letters for scalars and vectors, and script-case for sets e.g. $\mathcal{X}$. For vectors, such as $v \in \{0,1\}^{(\mathcal{I}\times\mathcal{J})\cup\mathcal{J}}$, where $\mathcal{I}$ and $\mathcal{J}$ are finite sets, we use the notation $v(i,j)$ and $v(j)$ to represent elements of the vector. Define $d = \delta(i)$ to be the indicator vector with $d(i) = 1$ and $d(i') = 0$ for all $i' \neq i$. Finally define the notation $[J]$ to refer to $\{1 \dots J\}$ and $[J]_0$ to refer to $\{0 \dots J\}$.

2 Background

The focus of this work is on the word alignment decoding problem. Given a sentence $\mathbf{e}$ of length $|\mathbf{e}| = I$ and a sentence $\mathbf{f}$ of length $|\mathbf{f}| = J$, our goal is to find the best bidirectional alignment between the two sentences under a given objective function. Before turning to the model of interest, we first introduce directional word alignment.

2.1 Word Alignment

In the $\mathbf{e}\rightarrow\mathbf{f}$ word alignment problem, each word in $\mathbf{e}$ is aligned to a word in $\mathbf{f}$ or to the null word ϵ. This alignment is a mapping from each index $i \in [I]$ to an index $j \in [J]_0$ (where $j = 0$ represents alignment to ϵ). We refer to a single word alignment as a *link*.

A first-order HMM alignment model (Vogel et al., 1996) is an HMM of length $I + 1$ where the hidden state at position $i \in [I]_0$ is the aligned index $j \in [J]_0$, and the transition score takes into account the previously aligned index $j' \in [J]_0$.[1] Formally, define the set of possible HMM alignments as $\mathcal{X} \subset \{0,1\}^{([I]_0\times[J]_0)\cup([I]\times[J]_0\times[J]_0)}$ with

$$\mathcal{X} = \begin{cases} x : x(0,0) = 1, \\ x(i,j) = \sum_{j'=0}^{J} x(j',i,j) & \forall i \in [I], j \in [J]_0, \\ x(i,j) = \sum_{j'=0}^{J} x(j,i+1,j') & \forall i \in [I-1]_0, j \in [J]_0 \end{cases}$$

where $x(i,j) = 1$ indicates that there is a link between index i and index j, and $x(j',i,j) = 1$ indicates that index $i-1$ aligns to index j' and index i aligns to j. Figure 1 shows an example member of $\mathcal{X}$.

The constraints of $\mathcal{X}$ enforce *backward* and *forward* consistency respectively. If $x(i,j) = 1$, backward consistency enforces that there is a transition from $(i-1,j')$ to (i,j) for some $j' \in [J]_0$, whereas forward consistency enforces a transition from (i,j) to $(i+1,j')$ for some $j' \in [J]_0$. Informally the constraints "chain" together the links.

The HMM objective function $f : \mathcal{X} \rightarrow \mathbb{R}$ can be written as a linear function of x

$$f(x;\theta) = \sum_{i=1}^{I}\sum_{j=0}^{J}\sum_{j'=0}^{J} \theta(j',i,j)x(j',i,j)$$

where the vector $\theta \in \mathbb{R}^{[I]\times[J]_0\times[J]_0}$ includes the transition and alignment scores. For a generative model of alignment, we might define $\theta(j',i,j) = \log(p(\mathbf{e}_i|\mathbf{f}_j)p(j|j'))$. For a discriminative model of alignment, we might define $\theta(j',i,j) = w \cdot \phi(i,j',j,\mathbf{f},\mathbf{e})$ for a feature function ϕ and weights w (Moore, 2005; Lacoste-Julien et al., 2006).

Now reverse the direction of the model and consider the $\mathbf{f}\rightarrow\mathbf{e}$ alignment problem. An $\mathbf{f}\rightarrow\mathbf{e}$ alignment is a binary vector $y \in \mathcal{Y}$ where for each $j \in [J]$, $y(i,j) = 1$ for exactly one $i \in [I]_0$. Define the set of HMM alignments $\mathcal{Y} \subset \{0,1\}^{([I]_0\times[J]_0)\cup([I]_0\times[I]_0\times[J])}$ as

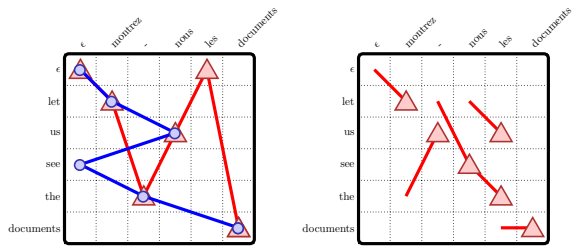
$$\mathcal{Y} = \begin{cases} y : y(0,0) = 1, \\ y(i,j) = \sum_{i'=0}^{I} y(i',i,j) & \forall i \in [I]_0, j \in [J], \\ y(i,j) = \sum_{i'=0}^{I} y(i,i',j+1) & \forall i \in [I]_0, j \in [J-1]_0 \end{cases}$$

Similarly define the objective function

$$g(y;\omega) = \sum_{j=1}^{J}\sum_{i=0}^{I}\sum_{i'=0}^{I} \omega(i',i,j)y(i',i,j)$$

with vector $\omega \in \mathbb{R}^{[I]_0\times[I]_0\times[J]}$.

[1] Our definition differs slightly from other HMM-based aligners in that it does not track the last ϵ alignment.



(a) (b)

Figure 2: (a) An example alignment pair (x, y) satisfying the full agreement conditions. The x alignment is represented with circles and the y alignment with triangles. (b) An example $\mathbf{f}\rightarrow\mathbf{e}$ alignment $y \in \mathcal{Y}'$ with relaxed forward constraints. Note that unlike an alignment from $\mathcal{Y}$ multiple words may be aligned in a column and words may transition from non-aligned positions.

Note that for both of these models we can solve the optimization problem exactly using the standard Viterbi algorithm for HMM decoding. The first can be solved in $O(IJ^2)$ time and the second in $O(I^2J)$ time.

3 Bidirectional Alignment

The directional bias of the $\mathbf{e}\rightarrow\mathbf{f}$ and $\mathbf{f}\rightarrow\mathbf{e}$ alignment models may cause them to produce differing alignments. To obtain the best single alignment, it is common practice to use a post-hoc algorithm to merge these directional alignments (Och et al., 1999). First, a directional alignment is found from each word in $\mathbf{e}$ to a word $\mathbf{f}$. Next an alignment is produced in the reverse direction from $\mathbf{f}$ to $\mathbf{e}$. Finally, these alignments are merged, either through intersection, union, or with an interpolation algorithm such as grow-diag-final (Koehn et al., 2003).

In this work, we instead consider a bidirectional alignment model that jointly considers both directional models. We begin in this section by introducing a simple bidirectional model that enforces *full* agreement between directional models and giving a relaxation for decoding. Section 4 loosens this model to *adjacent* agreement.

3.1 Enforcing Full Agreement

Perhaps the simplest post-hoc merging strategy is to retain the intersection of the two directional models. The analogous bidirectional model enforces *full* agreement to ensure the two alignments select the same non-null links i.e.

$$x^*, y^* = \arg\max_{x\in\mathcal{X}, y\in\mathcal{Y}} f(x) + g(y) \text{ s.t.}$$
$$x(i,j) = y(i,j) \quad \forall i \in [I], j \in [J]$$

We refer to the optimal alignments for this problem as x^* and y^*.

Unfortunately this bidirectional decoding model is NP-Hard (a proof is given in Appendix A). As it is common for alignment pairs to have $|\mathbf{f}|$ or $|\mathbf{e}|$ over 40, exact decoding algorithms are intractable in the worst-case.

Instead we will use Lagrangian relaxation for this model. At a high level, we will remove a subset of the constraints from the original problem and replace them with Lagrange multipliers. If we can solve this new problem efficiently, we may be able to get optimal solutions to the original problem. (See the tutorial by Rush and Collins (2012) describing the method.)

There are many possible subsets of constraints to consider relaxing. The relaxation we use preserves the agreement constraints while relaxing the Markov structure of the $\mathbf{f}\rightarrow\mathbf{e}$ alignment. This relaxation will make it simple to later re-introduce constraints in Section 5.

We relax the forward constraints of set $\mathcal{Y}$. Without these constraints the y links are no longer chained together. This has two consequences: (1) for index j there may be any number of indices i, such that $y(i,j) = 1$, (2) if $y(i',i,j) = 1$ it is no longer required that $y(i', j-1) = 1$. This gives a set $\mathcal{Y}'$ which is a superset of $\mathcal{Y}$

$$\mathcal{Y}' = \left\{ \begin{array}{l} y : y(0,0) = 1, \\ y(i,j) = \sum_{i'=0}^{I} y(i',i,j) \quad \forall i \in [I]_0, j \in [J] \end{array} \right.$$

Figure 2(b) shows a possible $y \in \mathcal{Y}'$ and a valid unchained structure.

To form the Lagrangian dual with relaxed forward constraints, we introduce a vector of Lagrange multipliers, $\lambda \in \mathbb{R}^{[I-1]_0 \times [J]_0}$, with one multiplier for each original constraint. The Lagrangian dual $L(\lambda)$ is defined as

$$\max_{\substack{x\in\mathcal{X}, y\in\mathcal{Y}', \\ x(i,j)=y(i,j)}} f(x) + \sum_{i=1}^{I}\sum_{j=0}^{J}\sum_{i'=0}^{I} y(i',i,j)\omega(i',i,j) \quad (1)$$
$$- \sum_{i=0}^{I}\sum_{j=0}^{J-1} \lambda(i,j)\left(y(i,j) - \sum_{i'=0}^{I} y(i,i',j+1)\right)$$
$$= \max_{\substack{x\in\mathcal{X}, y\in\mathcal{Y}', \\ x(i,j)=y(i,j)}} f(x) + \sum_{i=1}^{I}\sum_{j=0}^{J}\sum_{i'=0}^{I} y(i',i,j)\omega'(i',i,j) \quad (2)$$
$$= \max_{\substack{x\in\mathcal{X}, y\in\mathcal{Y}', \\ x(i,j)=y(i,j)}} f(x) + \sum_{i=1}^{I}\sum_{j=0}^{J} y(i,j) \max_{i'\in[I]_0} \omega'(i',i,j) \quad (3)$$
$$= \max_{\substack{x\in\mathcal{X}, y\in\mathcal{Y}', \\ x(i,j)=y(i,j)}} f(x) + g'(y;\omega,\lambda) \quad (4)$$

Line 2 distributes the λ's and introduces a modified potential vector ω' defined as

$$\omega'(i', i, j) = \omega(i', i, j) - \lambda(i, j) + \lambda(i', j-1)$$

for all $i' \in [I]_0, i \in [I]_0, j \in [J]$. Line 3 utilizes the relaxed set $\mathcal{Y}'$ which allows each $y(i,j)$ to select the best possible previous link $(i', j-1)$. Line 4 introduces the modified directional objective

$$g'(y; \omega, \lambda) = \sum_{i=1}^{I} \sum_{j=0}^{J} y(i,j) \max_{i' \in [I]_0} \omega'(i', i, j)$$

The Lagrangian dual is guaranteed to be an upper bound on the optimal solution, i.e. for all λ, $L(\lambda) \geq f(x^*) + g(y^*)$. Lagrangian relaxation attempts to find the tighest possible upper bound by minimizing the Lagrangian dual, $\min_\lambda L(\lambda)$, using subgradient descent. Briefly, subgradient descent is an iterative algorithm, with two steps. Starting with $\lambda = 0$, we iteratively

1. Set (x, y) to the $\arg\max$ of $L(\lambda)$.
2. Update $\lambda(i,j)$ for all $i \in [I-1]_0, j \in [J]_0$,

$$\lambda(i,j) \leftarrow \lambda(i,j) - \eta_t \big(y(i,j) - \sum_{i'=0}^{I} y(i, i', j+1)\big)$$

.

where $\eta_t > 0$ is a step size for the t'th update. If at any iteration of the algorithm the forward constraints are satisfied for (x, y), then $f(x) + g(y) = f(x^*) + g(x^*)$ and we say this gives a *certificate of optimality* for the underlying problem.

To run this algorithm, we need to be able to efficiently compute the (x, y) pair that is the $\arg\max$ of $L(\lambda)$ for any value of λ. Fortunately, since the y alignments are no longer constrained to valid transitions, we can compute these alignments by first picking the best $\mathbf{f} \rightarrow \mathbf{e}$ transitions for each possible link, and then running an $\mathbf{e} \rightarrow \mathbf{f}$ Viterbi-style algorithm to find the bidirectional alignment.

The $\max$ version of this algorithm is shown in Figure 3. It consists of two steps. We first compute the score for each $y(i,j)$ variable. We then use the standard Viterbi update for computing the x variables, adding in the score of the $y(i,j)$ necessary to satisfy the constraints.

procedure VITERBIFULL(θ, ω')
  Let π, ρ be dynamic programming charts.
  $\rho[i,j] \leftarrow \max_{i' \in [I]_0} \omega'(i', i, j) \; \forall \, i \in [I], j \in [J]_0$
  $\pi[0,0] \leftarrow \sum_{j=1}^{J} \max\{0, \rho[0,j]\}$
  for $i \in [I], j \in [J]_0$ in order **do**
    $\pi[i,j] \leftarrow \max_{j' \in [J]_0} \theta(j', i, j) + \pi[i-1, j']$
    if j $\neq 0$ **then** $\pi[i,j] \leftarrow \pi[i,j] + \rho[i,j]$
  return $\max_{j \in [J]_0} \pi[I,j]$

Figure 3: Viterbi-style algorithm for computing $L(\lambda)$. For simplicity the algorithm shows the $\max$ version of the algorithm, $\arg\max$ can be computed with back-pointers.

4 Adjacent Agreement

Enforcing full agreement can be too strict an alignment criteria. DeNero and Macherey (2011) instead propose a model that allows near matches, which we call *adjacent* agreement. Adjacent agreement allows links from one direction to agree with adjacent links from the reverse alignment for a small penalty. Figure 4(a) shows an example of a valid bidirectional alignment under adjacent agreement.

In this section we formally introduce adjacent agreement, and propose a relaxation algorithm for this model. The key algorithmic idea is to extend the Viterbi algorithm in order to consider possible adjacent links in the reverse direction.

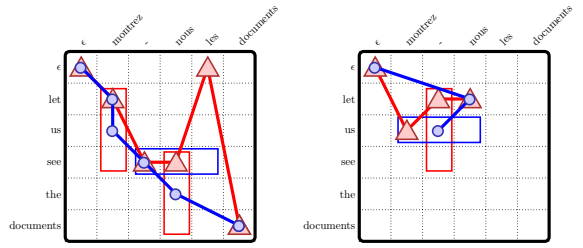
4.1 Enforcing Adjacency

Define the adjacency set $\mathcal{K} = \{-1, 0, 1\}$. A bidirectional alignment satisfies adjacency if for all $i \in [I], j \in [J]$,

- If $x(i,j) = 1$, it is required that $y(i+k, j) = 1$ for exactly one $k \in \mathcal{K}$ (i.e. either above, center, or below). We indicate which position with variables $z^{\updownarrow}_{i,j} \in \{0,1\}^{\mathcal{K}}$
- If $x(i,j) = 1$, it is allowed that $y(i, j+k) = 1$ for any $k \in \mathcal{K}$ (i.e. either left, center, or right) and all other $y(i, j') = 0$. We indicate which positions with variables $z^{\leftrightarrow}_{i,j} \in \{0,1\}^{\mathcal{K}}$

Formally for $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, the pair (x, y) is feasible if there exists a z from the set $\mathcal{Z}(x,y) \subset \{0,1\}^{\mathcal{K}^2 \times [I] \times [J]}$ defined as

$$\mathcal{Z}(x,y) = \left\{ \begin{array}{ll} z : \forall i \in [I], j \in [J] & \\ z^{\updownarrow}_{i,j} \in \{0,1\}^{\mathcal{K}}, & z^{\leftrightarrow}_{i,j} \in \{0,1\}^{\mathcal{K}} \\ x(i,j) = \sum_{k \in \mathcal{K}} z^{\updownarrow}_{i,j}(k), & \sum_{k \in \mathcal{K}} z^{\leftrightarrow}_{i,j}(k) = y(i,j), \\ z^{\updownarrow}_{i,j}(k) \leq y(i+k, j) & \forall k \in \mathcal{K} : i + k > 0, \\ x(i,j) \geq z^{\leftrightarrow}_{i,j-k}(k) & \forall k \in \mathcal{K} : j + k > 0 \end{array} \right.$$



(a) (b)

Figure 4: (a) An alignment satisfying the adjacency constraints. Note that $x(2,1) = 1$ is allowed because of $y(1,1) = 1$, $x(4,3) = 1$ because of $y(3,3)$, and $y(3,1)$ because of $x(3,2)$. (b) An adjacent bidirectional alignment in progress. Currently $x(2,2) = 1$ with $z^{\updownarrow}(-1) = 1$ and $z^{\leftrightarrow}(-1) = 1$. The last transition was from $x(1,3)$ with $z^{\leftrightarrow\prime}(-1) = 1, z^{\leftrightarrow\prime}(0) = 1, z^{\updownarrow\prime}(0) = 1$.

Additionally adjacent, non-overlapping matches are assessed a penalty α calculated as

$$h(z) = \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k \in \mathcal{K}} \alpha |k| (z^{\updownarrow}_{i,j}(k) + z^{\leftrightarrow}_{i,j}(k))$$

where $\alpha \leq 0$ is a parameter of the model. The example in Figure 4(a) includes a 3α penalty.

Adding these penalties gives the complete adjacent agreement problem

$$\arg\max_{\substack{z \in \mathcal{Z}(x,y) \\ x \in \mathcal{X}, y \in \mathcal{Y}}} f(x) + g(y) + h(z)$$

Next, apply the same relaxation from Section 3.1, i.e. we relax the forward constraints of the $\mathbf{f} \rightarrow \mathbf{e}$ set. This yields the following Lagrangian dual

$$L(\lambda) = \max_{\substack{z \in \mathcal{Z}(x,y) \\ x \in \mathcal{X}, y \in \mathcal{Y}'}} f(x) + g'(y; \omega, \lambda) + h(z)$$

Despite the new constraints, we can still compute $L(\lambda)$ in $O(IJ(I+J))$ time using a variant of the Viterbi algorithm. The main idea will be to consider possible adjacent settings for each link. Since each $z^{\updownarrow}_{i,j}$ and $z^{\leftrightarrow}_{i,j}$ only have a constant number of settings, this does not increase the asymptotic complexity of the algorithm.

Figure 5 shows the algorithm for computing $L(\lambda)$. The main loop of the algorithm is similar to Figure 3. It proceeds row-by-row, picking the best alignment $x(i,j) = 1$. The major change is that the chart π also stores a value $z \in \{0,1\}^{\mathcal{K} \times \mathcal{K}}$ representing a possible $z^{\updownarrow}_{i,j}, z^{\leftrightarrow}_{i,j}$ pair. Since we have

procedure VITERBIADJ(θ, ω')
$\rho[i,j] \leftarrow \max_{i' \in [I]_0} \omega'(i', i, j) \; \forall \, i \in [I], j \in [J]_0$
$\pi[0,0] \leftarrow \sum_{j=1}^{J} \max\{0, \rho[0,j]\}$
for $i \in [I], j \in [J]_0, z^{\updownarrow}, z^{\leftrightarrow} \in \{0,1\}^{|\mathcal{K}|}$ **do**
$\pi[i,j,z] \leftarrow$
$\max_{\substack{j' \in [J]_0, \\ z' \in \mathcal{N}(z, j-j')}} \theta(j', i, j) + \pi[i-1, j', z']$
$+ \sum_{k \in \mathcal{K}} z^{\leftrightarrow}(k)(\rho[i, j+k] + \alpha|k|)$
$+ z^{\updownarrow}(k)\alpha|k|$
return $\max_{j \in [J]_0, z \in \{0,1\}^{|\mathcal{K}| \times |\mathcal{K}|}} \pi[I, j, z]$

Figure 5: Modified Viterbi algorithm for computing the adjacent agreement $L(\lambda)$.

the proposed $z_{i,j}$ in the inner loop, we can include the scores of the adjacent y alignments that are in neighboring columns, as well as the possible penalty for matching $x(i,j)$ to a $y(i+k,j)$ in a different row. Figure 4(b) gives an example setting of z.

In the dynamic program, we need to ensure that the transitions between the z's are consistent. The vector z' indicates the y links adjacent to $x(i-1, j')$. If j' is near to j, z' may overlap with z and vice-versa. The transition set $\mathcal{N}$ ensures these indicators match up

$$\mathcal{N}(z, k') = \left\{ z' : \begin{array}{l} (z^{\updownarrow}(-1) \wedge k' \in \mathcal{K}) \Rightarrow z^{\leftrightarrow\prime}(k'), \\ (z^{\updownarrow\prime}(1) \wedge k' \in \mathcal{K}) \Rightarrow z^{\leftrightarrow}(-k'), \\ \sum_{k \in \mathcal{K}} z^{\updownarrow}(k) = 1 \end{array} \right.$$

5 Adding Back Constraints

In general, it can be shown that Lagrangian relaxation is only guaranteed to solve a linear programming relaxation of the underlying combinatorial problem. For difficult instances, we will see that this relaxation often does not yield provably exact solutions. However, it is possible to "tighten" the relaxation by re-introducing constraints from the original problem.

In this section, we extend the algorithm to allow incrementally re-introducing constraints. In particular we track which constraints are most often violated in order to explicitly enforce them in the algorithm.

Define a binary vector $p \in \{0,1\}^{[I-1]_0 \times [J]_0}$ where $p(i,j) = 1$ indicates a previously relaxed constraint on link $y(i,j)$ that should be reintroduced into the problem. Let the new partially

constrained Lagrangian dual be defined as

$$L(\lambda; p) = \max_{\substack{z \in \mathcal{Z}(x,y) \\ x \in \mathcal{X}, y \in \mathcal{Y}'}} f(x) + g'(y; \omega, \lambda) + h(z)$$

$$y(i,j) = \sum_{i'} y(i, i', j+1) \quad \forall i, j : p(i,j) = 1$$

If $p = \vec{1}$, the problem includes all of the original constraints, whereas $p = \vec{0}$ gives our original Lagrangian dual. In between we have progressively more constrained variants.

In order to compute the $\arg\max$ of this optimization problem, we need to satisfy the constraints within the Viterbi algorithm. We augment the Viterbi chart with a count vector $d \in \mathcal{D}$ where $\mathcal{D} \subset \mathbb{Z}^{||p||_1}$ and $d(i,j)$ is a count for the (i,j)'th constraint, i.e. $d(i,j) = y(i,j) - \sum_{i'} y(i', i, j)$. Only solutions with count 0 at the final position satisfy the active constraints. Additionally define a helper function $[\cdot]_{\mathcal{D}}$ as the projection from $\mathbb{Z}^{[I-1]_0 \times [J]} \rightarrow \mathcal{D}$, which truncates dimensions without constraints.

Figure 6 shows this constrained Viterbi relaxation approach. It takes p as an argument and enforces the active constraints. For simplicity, we show the full agreement version, but the adjacent agreement version is similar. The main new addition is that the inner loop of the algorithm ensures that the count vector d is the sum of the counts of its children d' and $d - d'$.

Since each additional constraint adds a dimension to d, adding constraints has a multiplicative impact on running time. Asymptotically the new algorithm requires $O(2^{||p||_1} IJ(I+J))$ time. This is a problem in practice as even adding a few constraints can make the problem intractable. We address this issue in the next section.

procedure CONSVITERBIFULL(θ, ω', p)
 for $i \in [I], j \in [J]_0, i' \in [I]$ **do**
  $d \leftarrow |\delta(i,j) - \delta(i', j-1)|_{\mathcal{D}}$
  $\rho[i,j,d] \leftarrow \omega'(i', i, j)$
 for $j \in [J], d \in \mathcal{D}$ **do**
  $\pi[0,0,d] \leftarrow \max_{d' \in \mathcal{D}} \pi[0,0,d'] + \rho[0,j,d-d']$
 for $i \in [I], j \in [J]_0, d \in \mathcal{D}$ **do**
  if j = 0 **then**
   $\pi[i,j,d] \leftarrow \max_{j' \in [J]_0} \theta(j', i, j) + \pi[i-1, j', d]$
  else
   $\pi[i,j,d] \leftarrow \max_{j' \in [J]_0, d' \in \mathcal{D}} \theta(j', i, j) + \pi[i-1, j', d'] + \rho[i, j, d-d']$
 return $\max_{j \in [J]_0} \pi[I, j, \mathbf{0}]$

Figure 6: Constrained Viterbi algorithm for finding partially-constrained, full-agreement alignments. The argument p indicates which constraints to enforce.

6 Pruning

Re-introducing constraints can lead to an exponential blow-up in the search space of the Viterbi algorithm. In practice though, many alignments in this space are far from optimal, e.g. aligning a common word like `the` to `nous` instead of `les`. Since Lagrangian relaxation re-computes the alignment many times, it would be preferable to skip these links in later rounds, particularly after re-introducing constraints.

In this section we describe an optimality preserving coarse-to-fine algorithm for pruning. Approximate coarse-to-fine pruning algorithms are widely used within NLP, but exact pruning is less common. Our method differs in that it only eliminates non-optimal transitions based on a lower-bound score. After introducing the pruning method, we present an algorithm to make this method effective in practice by producing high-scoring lower bounds for adjacent agreement.

6.1 Thresholding Max-Marginals

Our pruning method is based on removing transitions with low max-marginal values. Define the max-marginal value of an $\mathbf{e} \rightarrow \mathbf{f}$ transition in our Lagrangian dual as

$$M(j', i, j; \lambda) = \max_{\substack{z \in \mathcal{Z}(x,y), \\ x \in \mathcal{X}, y \in \mathcal{Y}'}} f(x) + g'(y; \lambda) + h(z) \quad \text{s.t. } x(j', i, j) = 1$$

where M gives the value of the best dual alignment that transitions from $(i-1, j')$ to (i, j). These max-marginals can be computed by running a forward-backward variant of any of the algorithms described thus far.

We make the following claim about max-marginal values and any lower-bound score

Lemma 1 (Safe Pruning). *For any valid constrained alignment $x \in \mathcal{X}, y \in \mathcal{Y}, z \in \mathcal{Z}(x,y)$ and for any dual vector $\lambda \in \mathbb{R}^{[I-1]_0 \times [J]_0}$, if there exists a transition j', i, j with max-marginal value $M(j', i, j; \lambda) < f(x) + g(y) + h(z)$ then the transition will not be in the optimal alignment, i.e. $x^*(j', i, j) = 0$.*

This lemma tells us that we can prune transitions whose dual max-marginal value falls below

a threshold without pruning possibly optimal transitions. Pruning these transitions can speed up Lagrangian relaxation without altering its properties.

Furthermore, the threshold is determined by any feasible lower bound on the optimal score, which means that better bounds can lead to more pruning.

6.2 Finding Lower Bounds

Since the effectiveness of pruning is dependent on the lower bound, it is crucial to be able to produce high-scoring alignments that satisfy the agreement constraints. Unfortunately, this problem is non-trivial. For instance, taking the union of directional alignments does not guarantee a feasible solution; whereas taking the intersection is trivially feasible but often not high-scoring.

To produce higher-scoring feasible bidirectional alignments we introduce a greedy heuristic algorithm. The algorithm starts with any feasible alignment (x, y, z). It runs the following greedy loop:

1. Repeat until there exists no $x(i, 0) = 1$ or $y(0, j) = 1$, or there is no score increase.
   (a) For each $i \in [I], j \in [J]_0, k \in \mathcal{K}$: $x(i, 0) = 1$, check if $x(i, j) \leftarrow 1$ and $y(i, j + k) \leftarrow 1$ is feasible, remember score.
   (b) For each $i \in [I]_0, j \in [J], k \in \mathcal{K}$: $y(0, j) = 1$, check if $y(i, j) \leftarrow 1$ and $x(i + k, j) \leftarrow 1$ is feasible, remember score.
   (c) Let (x, y, z) be the highest-scoring feasible solution produced.

This algorithm produces feasible alignments with monotonically increasing score, starting from the intersection of the alignments. It has run-time of $O(IJ(I + J))$ since each inner loop enumerates IJ possible updates and assigns at least one index a non-zero value, limiting the outer loop to $I + J$ iterations.

In practice we initialize the heuristic based on the intersection of x and y at the current round of Lagrangian relaxation. Experiments show that running this algorithm significantly improves the lower bound compared to just taking the intersection, and consequently helps pruning significantly.

7 Related Work

The most common techniques for bidirectional alignment are post-hoc combinations, such as union or intersection, of directional models, (Och et al., 1999), or more complex heuristic combiners such as grow-diag-final (Koehn et al., 2003).

Several authors have explored explicit bidirectional models in the literature. Cromieres and Kurohashi (2009) use belief propagation on a factor graph to train and decode a one-to-one word alignment problem. Qualitatively this method is similar to ours, although the model and decoding algorithm are different, and their method is not able to provide certificates of optimality.

A series of papers by Ganchev et al. (2010), Graca et al. (2008), and Ganchev et al. (2008) use posterior regularization to constrain the posterior probability of the word alignment problem to be symmetric and bijective. This work acheives state-of-the-art performance for alignment. Instead of utilizing posteriors our model tries to decode a single best one-to-one word alignment.

A different approach is to use constraints at training time to obtain models that favor bidirectional properties. Liang et al. (2006) propose agreement-based learning, which jointly learns probabilities by maximizing a combination of likelihood and agreement between two directional models.

General linear programming approaches have also been applied to word alignment problems. Lacoste-Julien et al. (2006) formulate the word alignment problem as quadratic assignment problem and solve it using an integer linear programming solver.

Our work is most similar to DeNero and Macherey (2011), which uses dual decomposition to encourage agreement between two directional HMM aligners during decoding time.

8 Experiments

Our experimental results compare the accuracy and optimality of our decoding algorithm to directional alignment models and previous work on this bidirectional model.

Data and Setup The experimental setup is identical to DeNero and Macherey (2011). Evaluation is performed on a hand-aligned subset of the NIST 2002 Chinese-English dataset (Ayan and Dorr, 2006). Following past work, the first 150 sentence pairs of the training section are used for evaluation. The potential parameters θ and ω are set based on unsupervised HMM models trained on the LDC FBIS corpus (6.2 million words).

	1-20 (28%)			21-40 (45%)			41-60 (27%)			all		
	time	cert	exact	time	cert	exact	time	cert	exact	time	cert	exact
ILP	15.12	100.0	100.0	364.94	100.0	100.0	2,829.64	100.0	100.0	924.24	100.0	100.0
LR	0.55	97.6	97.6	4.76	55.9	55.9	15.06	7.5	7.5	6.33	54.7	54.7
CONS	0.43	100.0	100.0	9.86	95.6	95.6	61.86	55.0	62.5	21.08	86.0	88.0
D&M	-	6.2	-	-	0.0	-	-	0.0	-	-	6.2	-

Table 1: Experimental results for model accuracy of bilingual alignment. Column *time* is the mean time per sentence pair in seconds; *cert* is the percentage of sentence pairs solved with a certificate of optimality; *exact* is the percentage of sentence pairs solved exactly. Results are grouped by sentence length. The percentage of sentence pairs in each group is shown in parentheses.

Training is performed using the agreement-based learning method which encourages the directional models to overlap (Liang et al., 2006). This directional model has been shown produce state-of-the-art results with this setup (Haghighi et al., 2009).

Baselines We compare the algorithm described in this paper with several baseline methods. DIR includes post-hoc combinations of the $\mathbf{e}\rightarrow\mathbf{f}$ and $\mathbf{f}\rightarrow\mathbf{e}$ HMM-based aligners. Variants include union, intersection, and grow-diag-final. D&M is the dual decomposition algorithm for bidirectional alignment as presented by DeNero and Macherey (2011) with different final combinations. LR is the Lagrangian relaxation algorithm applied to the adjacent agreement problem without the additional constraints described in Section 5. CONS is our full Lagrangian relaxation algorithm including incremental constraint addition. ILP uses a highly-optimized general-purpose integer linear programming solver to solve the lattice with the constraints described (Gurobi Optimization, 2013).

Implementation The main task of the decoder is to repeatedly compute the $\arg\max$ of $L(\lambda)$. To speed up decoding, our implementation fully instantiates the Viterbi lattice for a problem instance. This approach has several benefits: each iteration can reuse the same lattice structure; max-marginals can be easily computed with a general forward-backward algorithm; pruning corresponds to removing lattice edges; and adding constraints can be done through lattice intersection. For consistency, we implement each baseline (except for D&M) through the same lattice.

Parameter Settings We run 400 iterations of the subgradient algorithm using the rate schedule $\eta_t = 0.95^{t'}$ where t' is the count of updates for which the dual value did not improve. Every 10 iterations we run the greedy decoder to compute a lower bound. If the gap between our current dual value $L(\lambda)$ and the lower bound improves significantly we run coarse-to-fine pruning as described in Section 6 with the best lower bound. For CONS, if the algorithm does not find an optimal solution we run 400 more iterations and incrementally add the 5 most violated constraints every 25 iterations.

Model	Combiner	alignment			phrase pair		
		Prec	Rec	AER	Prec	Rec	F1
DIR	union	57.6	80.0	33.4	75.1	33.5	46.3
	intersection	86.2	62.9	27.0	64.3	43.5	51.9
	grow-diag	59.7	79.5	32.1	70.1	36.9	48.4
D&M	union	63.3	81.5	29.1	63.2	44.9	52.5
	intersection	77.5	75.1	23.6	57.1	53.6	55.3
	grow-diag	65.6	80.6	28.0	60.2	47.4	53.0
CONS		72.5	74.9	26.4	53.0	52.4	52.7

Table 2: Alignment accuracy and phrase pair extraction accuracy for directional and bidirectional models. `Prec` is the precision. `Rec` is the recall. `AER` is alignment error rate and `F1` is the phrase pair extraction F1 score.

Results Our first set of experiments looks at the model accuracy and the decoding time of various methods that can produce optimal solutions. Results are shown in Table 1. D&M is only able to find the optimal solution with certificate on 6% of instances. The relaxation algorithm used in this work is able to increase that number to 54.7%. With incremental constraints and pruning, we are able to solve over 86% of sentence pairs including many longer and more difficult pairs. Additionally the method finds these solutions with only a small increase in running time over Lagrangian relaxation, and is significantly faster than using an ILP solver.

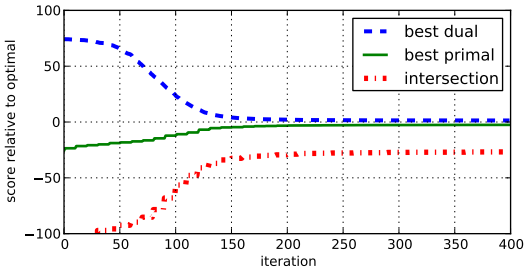
Next we compare the models in terms of alignment accuracy. Table 2 shows the precision, recall and alignment error rate (AER) for word alignment. We consider union, intersection and grow-diag-final as combination procedures. The combination procedures are applied to D&M in the case when the algorithm does not converge. For CONS, we use the optimal solution for the 86% of instances that converge and the highest-scoring greedy solution for those that do not. The proposed method has an AER of 26.4, which outperforms each of the directional models. However, although CONS achieves a higher model score than D&M, it performs worse in accuracy. Ta-

	1-20	21-40	41-60	all
# cons.	20.0	32.1	39.5	35.9

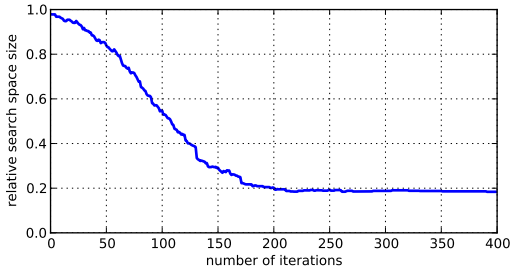
Table 3: The average number of constraints added for sentence pairs where Lagrangian relaxation is not able to find an exact solution.

ble 2 also compares the models in terms of phrase-extraction accuracy (Ayan and Dorr, 2006). We use the phrase extraction algorithm described by DeNero and Klein (2010), accounting for possible links and ϵ alignments. CONS performs better than each of the directional models, but worse than the best D&M model.

Finally we consider the impact of constraint addition, pruning, and use of a lower bound. Table 3 gives the average number of constraints added for sentence pairs for which Lagrangian relaxation alone does not produce a certificate. Figure 7(a) shows the average over all sentence pairs of the best dual and best primal scores. The graph compares the use of the greedy algorithm from Section 6.2 with the simple intersection of x and y. The difference between these curves illustrates the benefit of the greedy algorithm. This is reflected in Figure 7(b) which shows the effectiveness of coarse-to-fine pruning over time. On average, the pruning reduces the search space of each sentence pair to 20% of the initial search space after 200 iterations.



(a) The best dual and the best primal score, relative to the optimal score, averaged over all sentence pairs. The `best primal` curve uses a feasible greedy algorithm, whereas the `intersection` curve is calculated by taking the intersection of x and y.



(b) A graph showing the effectiveness of coarse-to-fine pruning. Relative search space size is the size of the pruned lattice compared to the initial size. The plot shows an average over all sentence pairs.

Figure 7

9 Conclusion

We have introduced a novel Lagrangian relaxation algorithm for a bidirectional alignment model that uses incremental constraint addition and coarse-to-fine pruning to find exact solutions. The algorithm increases the number of exact solution found on the model of DeNero and Macherey (2011) from 6% to 86%.

Unfortunately despite achieving higher model score, this approach does not produce more accurate alignments than the previous algorithm. This suggests that the adjacent agreement model may still be too constrained for this underlying task. Implicitly, an approach with fewer exact solutions may allow for useful violations of these constraints. In future work, we hope to explore bidirectional models with soft-penalties to explicitly permit these violations.

A Proof of NP-Hardness

We can show that the bidirectional alignment problem is NP-hard by reduction from the traveling salesman problem (TSP). A TSP instance with N cities has distance $c(i', i)$ for each $(i', i) \in [N]^2$. We can construct a sentence pair in which $I = J = N$ and ϵ-alignments have infinite cost.

$$\begin{aligned}
\omega(i', i, j) &= -c(i', i) && \forall i' \in [N]_0, i \in [N], j \in [N] \\
\theta(j', i, j) &= 0 && \forall j' \in [N]_0, i \in [N], j \in [N] \\
\omega(i', 0, j) &= -\infty && \forall i' \in [N]_0, j \in [N] \\
\theta(j', i, 0) &= -\infty && \forall j' \in [N]_0, i \in [N]
\end{aligned}$$

Every bidirectional alignment with finite objective score must align exactly one word in e to each word in f, encoding a permutation a. Moreover, each possible permutation has a finite score: the negation of the total distance to traverse the N cities in order a under distance c. Therefore, solving such a bidirectional alignment problem would find a minimal Hamiltonian path of the TSP encoded in this way, concluding the reduction.

Acknowledgments Alexander Rush, Yin-Wen Chang and Michael Collins were all supported by NSF grant IIS-1161814. Alexander Rush was partially supported by an NSF Graduate Research Fellowship.

References

Necip Fazil Ayan and Bonnie J Dorr. 2006. Going beyond aer: An extensive analysis of word alignments and their impact on mt. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 9–16. Association for Computational Linguistics.

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

Fabien Cromieres and Sadao Kurohashi. 2009. An alignment algorithm using belief propagation and a structure-based distortion model. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 166–174. Association for Computational Linguistics.

John DeNero and Dan Klein. 2010. Discriminative modeling of extraction sets for machine translation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1453–1463. Association for Computational Linguistics.

John DeNero and Klaus Macherey. 2011. Model-based aligner combination using dual decomposition. In *ACL*, pages 420–429.

Kuzman Ganchev, João V. Graça, and Ben Taskar. 2008. Better alignments = better translations? In *Proceedings of ACL-08: HLT*, pages 986–993, Columbus, Ohio, June. Association for Computational Linguistics.

K. Ganchev, J. Graça, J. Gillenwater, and B. Taskar. 2010. Posterior Regularization for Structured Latent Variable Models. *Journal of Machine Learning Research*, 11:2001–2049.

Joao Graca, Kuzman Ganchev, and Ben Taskar. 2008. Expectation maximization and posterior constraints. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 569–576. MIT Press, Cambridge, MA.

Inc. Gurobi Optimization. 2013. Gurobi optimizer reference manual.

Aria Haghighi, John Blitzer, John DeNero, and Dan Klein. 2009. Better word alignments with supervised itg models. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 923–931. Association for Computational Linguistics.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Simon Lacoste-Julien, Ben Taskar, Dan Klein, and Michael I Jordan. 2006. Word alignment via quadratic assignment. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 112–119. Association for Computational Linguistics.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 104–111. Association for Computational Linguistics.

Robert C Moore. 2005. A discriminative framework for bilingual word alignment. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 81–88. Association for Computational Linguistics.

Franz Josef Och, Christoph Tillmann, Hermann Ney, et al. 1999. Improved alignment models for statistical machine translation. In *Proc. of the Joint SIGDAT Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 20–28.

Alexander M Rush and Michael Collins. 2012. A tutorial on dual decomposition and lagrangian relaxation for inference in natural language processing. *Journal of Artificial Intelligence Research*, 45:305–362.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. Hmm-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*, pages 836–841. Association for Computational Linguistics.

A Recursive Recurrent Neural Network for Statistical Machine Translation

Shujie Liu[1], **Nan Yang**[2], **Mu Li**[1] **and Ming Zhou**[1]
[1]Microsoft Research Asia, Beijing, China
[2]University of Science and Technology of China, Hefei, China
shujliu, v-nayang, muli, mingzhou@microsoft.com

Abstract

In this paper, we propose a novel recursive recurrent neural network (R^2NN) to model the end-to-end decoding process for statistical machine translation. R^2NN is a combination of recursive neural network and recurrent neural network, and in turn integrates their respective capabilities: (1) new information can be used to generate the next hidden state, like recurrent neural networks, so that language model and translation model can be integrated naturally; (2) a tree structure can be built, as recursive neural networks, so as to generate the translation candidates in a bottom up manner. A semi-supervised training approach is proposed to train the parameters, and the phrase pair embedding is explored to model translation confidence directly. Experiments on a Chinese to English translation task show that our proposed R^2NN can outperform the state-of-the-art baseline by about 1.5 points in BLEU.

1 Introduction

Deep Neural Network (DNN), which essentially is a multi-layer neural network, has re-gained more and more attentions these years. With the efficient training methods, such as (Hinton et al., 2006), DNN is widely applied to speech and image processing, and has achieved breakthrough results (Kavukcuoglu et al., 2010; Krizhevsky et al., 2012; Dahl et al., 2012).

Applying DNN to natural language processing (NLP), representation or embedding of words is usually learnt first. Word embedding is a dense, low dimensional, real-valued vector. Each dimension of the vector represents a latent aspect of the word, and captures its syntactic and semantic properties (Bengio et al., 2006). Word embedding is usually learnt from large amount of monolingual corpus at first, and then fine tuned for special distinct tasks. Collobert et al. (2011) propose a multi-task learning framework with DNN for various NLP tasks, including part-of-speech tagging, chunking, named entity recognition, and semantic role labelling. Recurrent neural networks are leveraged to learn language model, and they keep the history information circularly inside the network for arbitrarily long time (Mikolov et al., 2010). Recursive neural networks, which have the ability to generate a tree structured output, are applied to natural language parsing (Socher et al., 2011), and they are extended to recursive neural tensor networks to explore the compositional aspect of semantics (Socher et al., 2013).

DNN is also introduced to Statistical Machine Translation (SMT) to learn several components or features of conventional framework, including word alignment, language modelling, translation modelling and distortion modelling. Yang et al. (2013) adapt and extend the CD-DNN-HMM (Dahl et al., 2012) method to HMM-based word alignment model. In their work, bilingual word embedding is trained to capture lexical translation information, and surrounding words are utilized to model context information. Auli et al. (2013) propose a joint language and translation model, based on a recurrent neural network. Their model predicts a target word, with an unbounded history of both source and target words. Liu et al. (2013) propose an additive neural network for SMT decoding. Word embedding is used as the input to learn translation confidence score, which is combined with commonly used features in the conventional log-linear model. For distortion modeling, Li et al. (2013) use recursive auto encoders to make full use of the entire merging phrase pairs, going beyond the boundary words with a maximum entropy classifier (Xiong et al., 2006).

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1491–1500,
Baltimore, Maryland, USA, June 23-25 2014.

Different from the work mentioned above, which applies DNN to components of conventional SMT framework, in this paper, we propose a novel R^2NN to model the end-to-end decoding process. R^2NN is a combination of recursive neural network and recurrent neural network. In R^2NN, new information can be used to generate the next hidden state, like recurrent neural networks, and a tree structure can be built, as recursive neural networks. To generate the translation candidates in a commonly used bottom-up manner, recursive neural networks are naturally adopted to build the tree structure. In recursive neural networks, all the representations of nodes are generated based on their child nodes, and it is difficult to integrate additional global information, such as language model and distortion model. In order to integrate these crucial information for better translation prediction, we combine recurrent neural networks into the recursive neural networks, so that we can use global information to generate the next hidden state, and select the better translation candidate.

We propose a three-step semi-supervised training approach to optimizing the parameters of R^2NN, which includes recursive auto-encoding for unsupervised pre-training, supervised local training based on the derivation trees of forced decoding, and supervised global training using early update strategy. So as to model the translation confidence for a translation phrase pair, we initialize the phrase pair embedding by leveraging the sparse features and recurrent neural network. The sparse features are phrase pairs in translation table, and recurrent neural network is utilized to learn a smoothed translation score with the source and target side information. We conduct experiments on a Chinese-to-English translation task to test our proposed methods, and we get about 1.5 BLEU points improvement, compared with a state-of-the-art baseline system.

The rest of this paper is organized as follows: Section 2 introduces related work on applying DNN to SMT. Our R^2NN framework is introduced in detail in Section 3, followed by our three-step semi-supervised training approach in Section 4. Phrase pair embedding method using translation confidence is elaborated in Section 5. We introduce our conducted experiments in Section 6, and conclude our work in Section 7.

2 Related Work

Yang et al. (2013) adapt and extend CD-DNN-HMM (Dahl et al., 2012) to word alignment. In their work, initial word embedding is firstly trained with a huge mono-lingual corpus, then the word embedding is adapted and fine tuned bilingually in a context-depended DNN HMM framework. Word embeddings capturing lexical translation information and surrounding words modeling context information are leveraged to improve the word alignment performance. Unfortunately, the better word alignment result generated by this model, cannot bring significant performance improvement on a end-to-end SMT evaluation task.

To improve the SMT performance directly, Auli et al. (2013) extend the recurrent neural network language model, in order to use both the source and target side information to scoring translation candidates. In their work, not only the target word embedding is used as the input of the network, but also the embedding of the source word, which is aligned to the current target word. To tackle the large search space due to the weak independence assumption, a lattice algorithm is proposed to rerank the n-best translation candidates, generated by a given SMT decoder.

Liu et al. (2013) propose an additive neural network for SMT decoding. RNNLM (Mikolov et al., 2010) is firstly used to generate the source and target word embeddings, which are fed into a one-hidden-layer neural network to get a translation confidence score. Together with other commonly used features, the translation confidence score is integrated into a conventional log-linear model. The parameters are optimized with development data set using mini-batch conjugate sub-gradient method and a regularized ranking loss.

DNN is also brought into the distortion modeling. Going beyond the previous work using boundary words for distortion modeling in BTG-based SMT decoder, Li et al. (2013) propose to apply recursive auto-encoder to make full use of the entire merged blocks. The recursive auto-encoder is trained with reordering examples extracted from word-aligned bilingual sentences. Given the representations of the smaller phrase pairs, recursive auto-encoder can generate the representation of the parent phrase pair with a re-ordering confidence score. The combination of reconstruction error and re-ordering error is used to be the objective function for the model training.

3 Our Model

In this section, we leverage DNN to model the end-to-end SMT decoding process, using a novel recursive recurrent neural network (R^{2}NN), which is different from the above mentioned work applying DNN to components of conventional SMT framework. R^{2}NN is a combination of recursive neural network and recurrent neural network, which not only integrates the conventional global features as input information for each combination, but also generates the representation of the parent node for the future candidate generation.

In this section, we briefly recall the recurrent neural network and recursive neural network in Section 3.1 and 3.2, and then we elaborate our R^{2}NN in detail in Section 3.3.

3.1 Recurrent Neural Network

Recurrent neural network is usually used for sequence processing, such as language model (Mikolov et al., 2010). Commonly used sequence processing methods, such as Hidden Markov Model (HMM) and n-gram language model, only use a limited history for the prediction. In HMM, the previous state is used as the history, and for n-gram language model (for example n equals to 3), the history is the previous two words. Recurrent neural network is proposed to use unbounded history information, and it has recurrent connections on hidden states, so that history information can be used circularly inside the network for arbitrarily long time.

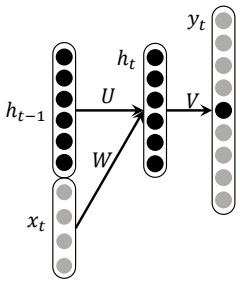


Figure 1: Recurrent neural network

As shown in Figure 1, the network contains three layers, an input layer, a hidden layer, and an output layer. The input layer is a concatenation of h_{t-1} and x_t, where h_{t-1} is a real-valued vector, which is the history information from time 0 to $t-1$. x_t is the embedding of the input word at time t. Word embedding x_t is integrated with previous history h_{t-1} to generate the current hidden layer, which is a new history vector h_t. Based on h_t, we can predict the probability of the next word, which forms the output layer y_t. The new history h_t is used for the future prediction, and updated with new information from word embedding x_t recurrently.

3.2 Recursive Neural Network

In addition to the sequential structure above, tree structure is also usually constructed in various NLP tasks, such as parsing and SMT decoding. To generate a tree structure, recursive neural networks are introduced for natural language parsing (Socher et al., 2011). Similar with recurrent neural networks, recursive neural networks can also use unbounded history information from the sub-tree rooted at the current node. The commonly used binary recursive neural networks generate the representation of the parent node, with the representations of two child nodes as the input.

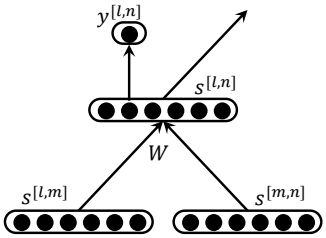


Figure 2: Recursive neural network

As shown in Figure 2, $s^{[l,m]}$ and $s^{[m,n]}$ are the representations of the child nodes, and they are concatenated into one vector to be the input of the network. $s^{[l,n]}$ is the generated representation of the parent node. $y_{[l,n]}$ is the confidence score of how plausible the parent node should be created. l, m, n are the indexes of the string. For example, for nature language parsing, $s^{[l,n]}$ is the representation of the parent node, which could be a NP or VP node, and it is also the representation of the whole sub-tree covering from l to n.

3.3 Recursive Recurrent Neural Network

Word embedding x_t is integrated as new input information in recurrent neural networks for each prediction, but in recursive neural networks, no additional input information is used except the two representation vectors of the child nodes. However, some global information , which cannot be generated by the child representations, is crucial

for SMT performance, such as language model score and distortion model score. So as to integrate such global information, and also keep the ability to generate tree structure, we combine the recurrent neural network and the recursive neural network to be a recursive recurrent neural network (R^{2}NN).

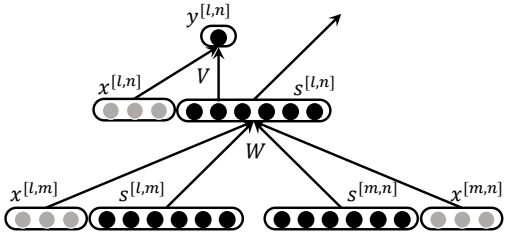


Figure 3: Recursive recurrent neural network

As shown in Figure 3, based on the recursive network, we add three input vectors $x^{[l,m]}$ for child node $[l,m]$, $x^{[m,n]}$ for child node $[m,n]$, and $x^{[l,n]}$ for parent node $[l,n]$. We call them recurrent input vectors, since they are borrowed from recurrent neural networks. The two recurrent input vectors $x^{[l,m]}$ and $x^{[m,n]}$ are concatenated as the input of the network, with the original child node representations $s^{[l,m]}$ and $s^{[m,n]}$. The recurrent input vector $x^{[l,n]}$ is concatenated with parent node representation $s^{[l,n]}$ to compute the confidence score $y^{[l,n]}$.

The input, hidden and output layers are calculated as follows:

$$\hat{x}^{[l,n]} = x^{[l,m]} \bowtie s^{[l,m]} \bowtie x^{[m,n]} \bowtie s^{[m,n]} \quad (1)$$

$$s_j^{[l,n]} = f(\sum_i \hat{x}_i^{[l,n]} w_{ji}) \quad (2)$$

$$y^{[l,n]} = \sum_j (s^{[l,n]} \bowtie x^{[l,n]})_j v_j \quad (3)$$

where $\bowtie$ is a concatenation operator in Equation 1 and Equation 3, and f is a non-linear function, here we use $HTanh$ function, which is defined as:

$$HTanh(x) = \begin{cases} -1, & x < -1 \\ x, & -1 \leq x \geq 1 \\ 1, & x > 1 \end{cases} \quad (4)$$

Figure 4 illustrates the R^{2}NN architecture for SMT decoding. For a source sentence "laizi faguo he eluosi de", we first split it into phrases "laizi", "faguo he eluosi" and "de". We then check whether translation candidates can be found in the translation table for each span, together with the phrase pair embedding and recurrent input vector (global features). We call it the rule matching phase. For a translation candidate of the span node $[l,m]$, the black dots stand for the node representation $s^{[l,m]}$, while the grey dots for recurrent input vector $x^{[l,m]}$. Given $s^{[l,m]}$ and $x^{[l,m]}$ for matched translation candidates, conventional CKY decoding process is performed using R^{2}NN. R^{2}NN can combine the translation pairs of child nodes, and generate the translation candidates for parent nodes with their representations and plausible scores. Only the n-best translation candidates are kept for upper combination, according to their plausible scores.

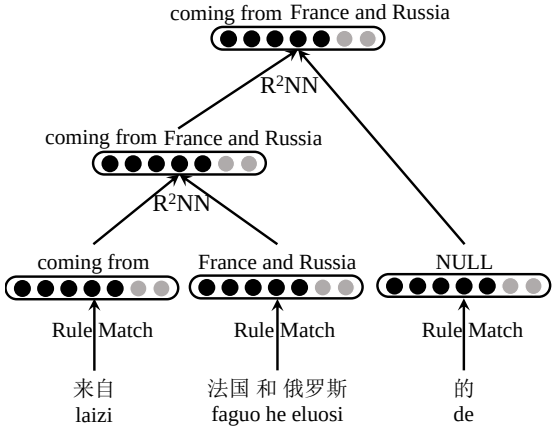


Figure 4: R^{2}NN for SMT decoding

We extract phrase pairs using the conventional method (Och and Ney, 2004). The commonly used features, such as translation score, language model score and distortion score, are used as the recurrent input vector x. During decoding, recurrent input vectors x for internal nodes are calculated accordingly. The difference between our model and the conventional log-linear model includes:

- R^{2}NN is not linear, while the conventional model is a linear combination.
- Representations of phrase pairs are automatically learnt to optimize the translation performance, while features used in conventional model are hand-crafted.
- History information of the derivation can be recorded in the representation of internal nodes, while conventional model cannot.

Liu et al. (2013) apply DNN to SMT decoding, but not in a recursive manner. A feature is learnt via a one-hidden-layer neural network, and the embedding of words in the phrase pairs are used as the input vector. Our model generates the representation of a translation pair based on its child nodes. Li et al. (2013) also generate the representation of phrase pairs in a recursive way. In their work, the representation is optimized to learn a distortion model using recursive neural network, only based on the representation of the child nodes. Our R^{2}NN is used to model the end-to-end translation process, with recurrent global information added. We also explore phrase pair embedding method to model translation confidence directly, which is introduced in Section 5.

In the next two sections, we will answer the following questions: (a) how to train the model, and (b) how to generate the initial representations of translation pairs.

4 Model Training

In this section, we propose a three-step training method to train the parameters of our proposed R^{2}NN, which includes unsupervised pre-training using recursive auto-encoding, supervised local training on the derivation tree of forced decoding, and supervised global training using early update training strategy.

4.1 Unsupervised Pre-training

We adopt the Recursive Auto Encoding (RAE) (Socher et al., 2011) for our unsupervised pre-training. The main idea of auto encoding is to initialize the parameters of the neural network, by minimizing the information lost, which means, capturing as much information as possible in the hidden states from the input vector.

As shown in Figure 5, RAE contains two parts, an encoder with parameter W, and a decoder with parameter W'. Given the representations of child nodes s_1 and s_2, the encoder generates the representation of parent node s. With the parent node representation s as the input vector, the decoder reconstructs the representation of two child nodes s_1' and s_2'. The loss function is defined as following so as to minimize the information lost:

$$L_{RAE}(s_1, s_2) = \frac{1}{2}(\|s_1 - s_1'\|^2 + \|s_2 - s_2'\|^2) \quad (5)$$

where $\|\cdot\|$ is the Euclidean norm.

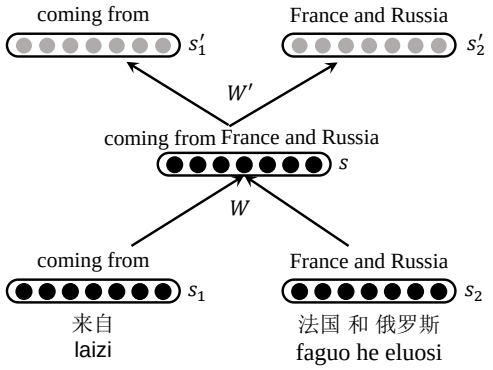


Figure 5: Recursive auto encoding for unsupervised pre-training

The training samples for RAE are phrase pairs $\{s_1, s_2\}$ in translation table, where s_1 and s_2 can form a continuous partial sentence pair in the training data. When RAE training is done, only the encoding model W will be fine tuned in the future training phases.

4.2 Supervised Local Training

We use contrastive divergence method to fine tune the parameters W and V. The loss function is the commonly used ranking loss with a margin, and it is defined as follows:

$$L_{SLT}(W, V, s^{[l,n]}) = max(0, 1 - y_{oracle}^{[l,n]} + y_t^{[l,n]}) \quad (6)$$

where $s^{[l,n]}$ is the source span. $y_{oracle}^{[l,n]}$ is the plausible score of a oracle translation result. $y_t^{[l,n]}$ is the plausible score for the best translation candidate given the model parameters W and V. The loss function aims to learn a model which assigns the good translation candidate (the oracle candidate) higher score than the bad ones, with a margin 1.

Translation candidates generated by forced decoding (Wuebker et al., 2010) are used as oracle translations, which are the positive samples. Forced decoding performs sentence pair segmentation using the same translation system as decoding. For each sentence pair in the training data, SMT decoder is applied to the source side, and any candidate which is not the partial sub-string of the target sentence is removed from the n-best list during decoding. From the forced decoding result, we can get the ideal derivation tree in the decoder's search space, and extract positive/oracle translation candidates.

4.3 Supervised Global Training

The supervised local training uses the n-odes/samples in the derivation tree of forced decoding to update the model, and the trained model tends to over-fit to local decisions. In this subsection, a supervised global training is proposed to tune the model according to the final translation performance of the whole source sentence.

Actually, we can update the model from the root of the decoding tree and perform back propagation along the tree structure. Due to the inexact search nature of SMT decoding, search errors may inevitably break theoretical properties, and the final translation results may be not suitable for model training. To handle this problem, we use early update strategy for the supervised global training. Early update is testified to be useful for SMT training with large scale features (Yu et al., 2013). Instead of updating the model using the final translation results, early update approach optimizes the model, when the oracle translation candidate is pruned from the n-best list, meaning that, the model is updated once it performs a unrecoverable mistake. Back propagation is performed along the tree structure, and the phrase pair embeddings of the leaf nodess are updated.

The loss function for supervised global training is defined as follows:

$$L_{SGT}(W, V, s^{[l,n]}) = -\log(\frac{\sum_{y^{[l,n]}_{oracle}} \exp(y^{[l,n]}_{oracle})}{\sum_{t \in nbest} \exp(y^{[l,n]}_{t})}) \quad (7)$$

where $y^{[l,n]}_{oracle}$ is the model score of a oracle translation candidate for the span $[l, n]$. Oracle translation candidates are candidates get from forced decoding. If the span $[l, n]$ is not the whole source sentence, there may be several oracle translation candidates, otherwise, there is only one, which is exactly the target sentence. There are much fewer training samples than those for supervised local training, and it is not suitable to use ranking loss for global training any longer. We use negative log-likelihood to penalize all the other translation candidates except the oracle ones, so as to leverage all the translation candidates as training samples.

5 Phrase Pair Embedding

The next question is how to initialize the phrase pair embedding in the translation table, so as to generate the leaf nodes of the derivation tree. There are more phrase pairs than mono-lingual words, but bilingual corpus is much more difficult to acquire, compared with monolingual corpus.

Embedding	#Data	#Entry	#Parameter
Word	1G	500K	$20 \times 500K$
Word Pair	7M	$(500K)^2$	$20 \times (500K)^2$
Phrase Pair	7M	$(500K)^4$	$20 \times (500K)^4$

Table 1: The relationship between the size of training data and the number of model parameters. The numbers for word embedding is calculated on English Giga-Word corpus version 3. For word pair and phrase pair embedding, the numbers are calculated on IWSLT 2009 dialog training set. The word count of each side of phrase pairs is limited to be 2.

Table 1 shows the relationship between the size of training data and the number of model parameters. For word embedding, the training size is 1G bits, and we may have 500K terms. For each term, we have a vector with length 20 as parameters, so there are $20 \times 500K$ parameters totally. But for source-target word pair, we may only have 7M bilingual corpus for training (taking IWSLT data set as an example), and there are $20 \times (500K)^2$ parameters to be tuned. For phrase pairs, the situation becomes even worse, especially when the limitation of word count in phrase pairs is relaxed. It is very difficult to learn the phrase pair embedding brute-forcedly as word embedding is learnt (Mikolov et al., 2010; Collobert et al., 2011), since we may not have enough training data.

A simple approach to construct phrase pair embedding is to use the average of the embeddings of the words in the phrase pair. One problem is that, word embedding may not be able to model the translation relationship between source and target phrases at phrase level, since some phrases cannot be decomposed. For example, the meaning of "hot dog" is not the composition of the meanings of the words "hot" and "dog". In this section, we split the phrase pair embedding into two parts to model the translation confidence directly: translation confidence with sparse features and translation confidence with recurrent neural network. We first get two translation confidence vectors separately using sparse features and recurrent neural network, and then concatenate them to be the phrase pair embedding. We call it translation confidence based phrase pair embedding (TCBPPE).

5.1 Translation Confidence with Sparse Features

Large scale feature training has drawn more attentions these years (Liang et al., 2006; Yu et al., 2013). Instead of integrating the sparse features directly into the log-linear model, we use them as the input to learn a phrase pair embedding. For the top 200,000 frequent translation pairs, each of them is a feature in itself, and a special feature is added for all the infrequent ones.

The one-hot representation vector is used as the input, and a one-hidden-layer network generates a confidence score. To train the neural network, we add the confidence scores to the conventional log-linear model as features. Forced decoding is utilized to get positive samples, and contrastive divergence is used for model training. The neural network is used to reduce the space dimension of sparse features, and the hidden layer of the network is used as the phrase pair embedding. The length of the hidden layer is empirically set to 20.

5.2 Translation Confidence with Recurrent Neural Network

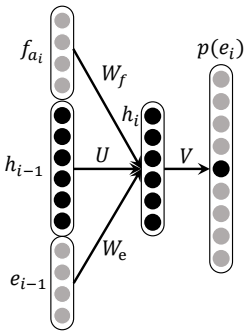


Figure 6: Recurrent neural network for translation confidence

We use recurrent neural network to generate two smoothed translation confidence scores based on source and target word embeddings. One is source to target translation confidence score and the other is target to source. These two confidence scores are defined as:

$$T_{S2T}(s,t) = \sum_i \log p(e_i|e_{i-1}, f_{a_i}, h_i) \quad (8)$$

$$T_{T2S}(s,t) = \sum_j \log p(f_j|f_{j-1}, e_{\hat{a}_j}, h_j) \quad (9)$$

where, f_{a_i} is the corresponding target word aligned to e_i, and it is similar for $e_{\hat{a}_j}$. $p(e_i|e_{i-1}, f_{a_i}, h_i)$ is produced by a recurrent network as shown in Figure 6. The recurrent neural network is trained with word aligned bilingual corpus, similar as (Auli et al., 2013).

6 Experiments and Results

In this section, we conduct experiments to test our method on a Chinese-to-English translation task. The evaluation method is the case insensitive IBM BLEU-4 (Papineni et al., 2002). Significant testing is carried out using bootstrap re-sampling method proposed by (Koehn, 2004) with a 95% confidence level.

6.1 Data Setting and Baseline

The data is from the IWSLT 2009 dialog task. The training data includes the BTEC and SLDB training data. The training data contains 81k sentence pairs, 655K Chinese words and 806K English words. The language model is a 5-gram language model trained with the target sentences in the training data. The test set is development set 9, and the development set comprises both development set 8 and the Chinese DIALOG set.

The training data for monolingual word embedding is Giga-Word corpus version 3 for both Chinese and English. Chinese training corpus contains 32M sentences and 1.1G words. English training data contains 8M sentences and 247M terms. We only train the embedding for the top 100,000 frequent words following (Collobert et al., 2011). With the trained monolingual word embedding, we follow (Yang et al., 2013) to get the bilingual word embedding using the IWSLT bilingual training data.

Our baseline decoder is an in-house implementation of Bracketing Transduction Grammar (BTG) (Wu, 1997) in CKY-style decoding with a lexical reordering model trained with maximum entropy (Xiong et al., 2006). The features of the baseline are commonly used features as standard BTG decoder, such as translation probabilities, lexical weights, language model, word penalty and distortion probabilities. All these commonly used features are used as recurrent input vector x in our R^{2}NN.

6.2 Translation Results

As we mentioned in Section 5, constructing phrase pair embeddings from word embeddings may be not suitable. Here we conduct experiments to ver-

ify it. We first train the source and target word embeddings separately using large monolingual data, following (Collobert et al., 2011). Using monolingual word embedding as the initialization, we fine tune them to get bilingual word embedding (Yang et al., 2013).

The word embedding based phrase pair embedding (WEPPE) is defined as:

$$Epp_{web}(s,t) = \sum_i E_{wms}(s_i) \bowtie \sum_j E_{wbs}(s_j) \bowtie \sum_k E_{wmt}(t_k) \bowtie \sum_l E_{wbt}(t_l) \quad (10)$$

where $\bowtie$ is a concatenation operator. s and t are the source and target phrases. $E_{wms}(s_i)$ and $E_{wmt}(t_k)$ are the monolingual word embeddings, and $E_{wbs}(s_i)$ and $E_{wbt}(t_k)$ are the bilingual word embeddings. Here the length of the word embedding is also set to 20. Therefore, the length of the phrase pair embedding is $20 \times 4 = 80$.

We compare our phrase pair embedding methods and our proposed R^{2}NN with baseline system, in Table 2. We can see that, our R^{2}NN models with WEPPE and TCBPPE are both better than the baseline system. WEPPE cannot get significant improvement, while TCBPPE does, compared with the baseline result. TCBPPE is much better than WEPPE.

Setting	Development	Test
Baseline	46.81	39.29
WEPPE+R^{2}NN	47.23	39.92
TCBPPE+R^{2}NN	48.70 $\uparrow$	40.81 $\uparrow$

Table 2: Translation results of our proposed R^{2}NN Model with two phrase embedding methods, compared with the baseline. Setting "WEPPE+R^{2}NN" is the result with word embedding based phrase pair embedding and our R^{2}NN Model, and "TCBPPE+R^{2}NN" is the result of translation confidence based phrase pair embedding and our R^{2}NN Model. The results with $\uparrow$ are significantly better than the baseline.

Word embedding can model translation relationship at word level, but it may not be powerful to model the phrase pair respondents at phrasal level, since the meaning of some phrases cannot be decomposed into the meaning of words. And also, translation task is difference from other NLP tasks, that, it is more important to model the translation confidence directly (the confidence of one target phrase as a translation of the source phrase), and our TCBPPE is designed for such purpose.

6.3 Effects of Global Recurrent Input Vector

In order to compare R^{2}NN with recursive network for SMT decoding, we remove the recurrent input vector in R^{2}NN to test its effect, and the results are shown in Table 3. Without the recurrent input vectors, R^{2}NN degenerates into recursive neural network (RNN).

Setting	Development	Test
WEPPE+R^{2}NN	47.23	40.81
WEPPE+RNN	37.62	33.29
TCBPPE+R^{2}NN	48.70	40.81
TCBPPE+RNN	45.11	37.33

Table 3: Experimental results to test the effects of recurrent input vector. WEPPE /TCBPPE+RNN are the results removing recurrent input vectors with WEPPE /TCBPPE.

From Table 3 we can find that, the recurrent input vector is essential to SMT performance. When we remove it from R^{2}NN, WEPPE based method drops about 10 BLEU points on development data and more than 6 BLEU points on test data. TCBPPE based method drops about 3 BLEU points on both development and test data sets. When we remove the recurrent input vectors, the representations of recursive network are generated with the child nodes, and it does not integrate global information, such as language model and distortion model, which are crucial to the performance of SMT.

6.4 Sparse Features and Recurrent Network Features

To test the contributions of sparse features and recurrent network features, we first remove all the recurrent network features to train and test our R^{2}NN model, and then remove all the sparse features to test the contribution of recurrent network features.

Setting	Development	Test
TCBPPE+R^{2}NN	48.70	40.81
SF+R^{2}NN	48.23	40.19
RNN+R^{2}NN	47.89	40.01

Table 4: Experimental results to test the effects of sparse features and recurrent network features.

The results are shown in Table 6.4. From the results, we can find that, sparse features are more effective than the recurrent network features a little bit. The sparse features can directly model the translation correspondence, and they may be more effective to rank the translation candidates, while recurrent neural network features are smoothed lexical translation confidence.

7 Conclusion and Future Work

In this paper, we propose a Recursive Recurrent Neural Network(R^2NN) to combine the recurrent neural network and recursive neural network. Our proposed R^2NN cannot only integrate global input information during each combination, but also can generate the tree structure in a recursive way. We apply our model to SMT decoding, and propose a three-step semi-supervised training method. In addition, we explore phrase pair embedding method, which models translation confidence directly. We conduct experiments on a Chinese-to-English translation task, and our method outperforms a state-of-the-art baseline about 1.5 points BLEU.

From the experiments, we find that, phrase pair embedding is crucial to the performance of SMT. In the future, we will explore better methods for phrase pair embedding to model the translation equivalent between source and target phrases. We will apply our proposed R^2NN to other tree structure learning tasks, such as natural language parsing.

References

Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint language and translation modeling with recurrent neural networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1044–1054, Seattle, Washington, USA, October. Association for Computational Linguistics.

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. *Innovations in Machine Learning*, pages 137–186.

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

George E Dahl, Dong Yu, Li Deng, and Alex Acero. 2012. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1):30–42.

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

Koray Kavukcuoglu, Pierre Sermanet, Y-Lan Boureau, Karol Gregor, Michaël Mathieu, and Yann LeCun. 2010. Learning convolutional feature hierarchies for visual recognition. *Advances in Neural Information Processing Systems*, pages 1090–1098.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 388–395.

Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1106–1114.

Peng Li, Yang Liu, and Maosong Sun. 2013. Recursive autoencoders for ITG-based translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 567–577, Seattle, Washington, USA, October. Association for Computational Linguistics.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 761–768. Association for Computational Linguistics.

Lemao Liu, Taro Watanabe, Eiichiro Sumita, and Tiejun Zhao. 2013. Additive neural networks for statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 791–801, Sofia, Bulgaria, August. Association for Computational Linguistics.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of the Annual Conference of International Speech Communication Association*, pages 1045–1048.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational linguistics*, 30(4):417–449.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Richard Socher, Cliff C Lin, Andrew Y Ng, and Christopher D Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, volume 2, page 7.

Richard Socher, John Bauer, and Christopher D Manning. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 455–465.

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

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

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, volume 44, page 521.

Nan Yang, Shujie Liu, Mu Li, Ming Zhou, and Nenghai Yu. 2013. Word alignment modeling with context dependent deep neural network. In *51st Annual Meeting of the Association for Computational Linguistics*.

Heng Yu, Liang Huang, Haitao Mi, and Kai Zhao. 2013. Max-violation perceptron and forced decoding for scalable MT training. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1112–1123, Seattle, Washington, USA, October. Association for Computational Linguistics.

Predicting Instructor's Intervention in MOOC forums

Snigdha Chaturvedi **Dan Goldwasser** **Hal Daumé III**
Department of Computer Science,
University of Maryland, College Park, Maryland
{snigdhac, goldwas1, hal}@umiacs.umd.edu

Abstract

Instructor intervention in student discussion forums is a vital component in Massive Open Online Courses (MOOCs), where personalized interaction is limited. This paper introduces the problem of predicting instructor interventions in MOOC forums. We propose several prediction models designed to capture unique aspects of MOOCs, combining course information, forum structure and posts content. Our models abstract contents of individual posts of threads using latent categories, learned jointly with the binary intervention prediction problem. Experiments over data from two *Coursera* MOOCs demonstrate that incorporating the structure of threads into the learning problem leads to better predictive performance.

1 Introduction

Ubiquitous computing and easy access to high bandwidth internet have reshaped the *modus operandi* in distance education towards Massive Open Online Courses (MOOCs). Courses offered by ventures such as Coursera and Udacity now impart inexpensive and high-quality education from field-experts to thousands of learners across geographic and cultural barriers.

Even as the MOOC model shows exciting possibilities, it presents a multitude of challenges that must first be negotiated to completely realize its potential. MOOCs platforms have been especially criticized on grounds of lacking a personalized educational experience (Edmundson, 2012). Unlike traditional classrooms, the predominant mode of interaction between students and instructors in MOOCs is via online discussion forums. Ideally, forum discussions can help make up for the lack of direct interaction, by enabling students to ask questions and clarify doubts. However, due to huge class sizes, even during the short duration of a course, MOOCs witness a very large number of threads on these forums. Owing to extremely skewed ratios of students to instructional staff, it can be prohibitively time-consuming for the instructional staff to manually follow all threads of a forum. Hence there is a pressing need for automatically *curating* the discussions for the instructors.

In this paper, we focus on identifying situations in which instructor (used interchangeably with "instructional staff" in this paper) intervention is warranted. Using existing forum posts and interactions, we frame this as a binary prediction problem of identifying instructor's intervention in forum threads. Our initial analysis revealed that instructors usually intervene on threads discussing students' issues close to a quiz or exam. They also take interest in grading issues and logistics problems. There are multiple cues specific to the MOOC setting, which when combined with the rich lexical information present in the forums, can yield useful predictive models.

Analyzing forum-postings contents and bringing the most pertinent content to the instructor's attention would help instructors receive timely feedback and design interventions as needed. From the students' perspective, the problem is evident from an examination of existing forum content, indicating that if students want instructor's input on some issues, the only way for them to get his/her attention is by 'up-voting' their votes. Fig. 1 provides some examples of this behavior. This is clearly an inefficient solution.

Our main technical contribution is introducing three different models addressing the task of predicting instructor interventions. The first uses a logistic regression model that primarily incorporates high level information about threads and posts. However, forum threads have structure which is not leveraged our initial model. We present two

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1501–1511,
Baltimore, Maryland, USA, June 23-25 2014.

"The problem summary: Anyone else having problems viewing the video lecture...very choppy. If you are also experiencing this issue; please upvote this post."
"I read that by up-voting threads and posts you can get the instructors' attention faster."
"Its is very bad to me that I achieved 10 marks in my 1st assignment and now 9 marks in my 2nd assignment, now I won't get certificate, please Course staff it is my appeal to change the passing scheme or please be lenient. Please upvote my post so that staff take this problem under consideration."

Figure 1: Sample posts that showing students desiring instructor's attention have to resolve to the inefficient method of getting their posts upvoted.

additional structured models. Both models assume that posts of a thread structure it in form of a story or a "chain of events." For example, an opening post of a thread might pose a question and the following posts can then answer or comment on the question. Our second and third models tap this linear 'chain of events' behavior by assuming that individual posts belong to latent categories which represent their textual content at an abstract level and that an instructor's decision to reply to a post is based on this chain of events (represented by the latent categories). We present two different ways of utilizing this 'chain of events' behavior for predicting instructor's intervention which can be either simply modeled as the 'next step' is this chain of events (*Linear Chain Markov Model*) or as a decision globally depending on the entire chain (*Global Chain Model*). Our experiments on two different datasets reveal that using the latent post categories helps in better prediction.

Our contributions can be summarized as:

- We motivate and introduce the important problem of predicting instructor intervention in MOOC forums
- We present two chain based models that incorporate thread structure.
- We show the utility of modeling thread structure, and the value of lexical and domain specific knowledge for the prediction task

2 Related Work

To the best of our knowledge, the problem of predicting instructor's intervention in MOOC forums has not been addressed yet. Prior work deals with analyzing general online discussion forums of social media sites (Kleinberg, 2013): such as predicting comment volume (Backstrom et al., 2013; De Choudhury et al., 2009; Wang et al., 2012; Tsagkias et al., 2009; Yano and Smith, 2010; Artzi et al., 2012) and rate of content diffusion (Kwak et al., 2010; Lerman and Ghosh, 2010; Bakshy et al., 2011; Romero et al., 2011; Artzi et al., 2012) and also question answering (Chaturvedi et al., 2014).

Wang et al. (2007) incorporate thread structure of conversations using features in email threads while Goldwasser and Daumé III (2014) use latent structure, aimed to identify relevant dialog segments, for predicting objections during courtroom deliberations. Other related work include speech act recognition in emails and forums but at a sentence level (Jeong et al., 2009), and using social network analysis to improve message classification into pre-determined types (Fortuna et al., 2007). Discussion forums data has also been used to address other interesting challenges such as extracting chatbox knowledge for use in general online forums (Huang et al., 2007) and automatically extracting answers from discussion forums (Catherine et al., 2013), subjectivity analysis of online forums (Biyani et al., 2013). Most of these methods use ideas similar to ours: identifying that threads (or discussions) have an underlying structure and that messages belong to categories. However, they operate in a different domain, which makes their goals and methods different from ours.

Our work is most closely related to that of Backstrom et al. (2013) which introduced the re-entry prediction task —predicting whether a user who has participated in a thread will later contribute another comment to it. While seemingly related, their prediction task, focusing on users who have already commented on a thread, and their algorithmic approach are different than ours. Our work is also very closely related to that of Wang et al. (2013) who predict solvedness —which predicts if there is a solution to the original problem posted in the thread. Like us, they believe that category of posts can assist in the prediction task, however, possibly owing to the complexity of general discussion forums, they had to manually create and annotate data with a sophisticated taxonomy. We do not make such assumptions.

The work presented in (Gómez et al., 2008;

Liben-Nowell and Kleinberg, 2008; Kumar et al., 2010; Golub and Jackson, 2010; Wang et al., 2011; Aumayr et al., 2011) discuss characterizing threads using reply-graphs (often trees) and learning this structure. However, this representation is not natural for the MOOC domain where discussions are relatively more focused on the thread topic and are better organized using sections within the forums.

Although most prior work focuses on discussion forums of social media sites such as Twitter or Facebook, where the dynamics of interaction is very different from MOOCs, a small number of recent work address the unique MOOC setting.

Stump et al. (2013) propose a framework for categorizing forum posts by designing a taxonomy and annotating posts manually to assist general forum analysis. Our model learns categories in a data-driven manner guided by the binary supervision (intervention decision) and serves a different purpose. Nevertheless, in Sec. 4.3 we compare the categories learnt by our models with those proposed by Stump et al. (2013).

Apart from this, recent works have looked into interesting challenges in this domain such as better peer grading models (Piech et al., 2013), code review (Huang et al., 2013; Nguyen et al., 2014), improving student engagement (Anderson et al., 2014) and understanding how students learn and code (Piech et al., 2012; Kizilcec et al., 2013; Ramesh et al., 2013).

3 Intervention Prediction Models

In this section, we explain our models in detail.

3.1 Problem Setting

In our description it is assumed that a discussion board is organized into multiple forums (representing topics such as "Assignment", "Study Group" etc.). A forum consists of multiple threads. Each thread (t) has a title and consists of multiple posts (p_i). Individual posts do not have a title and the number of posts varies dramatically from one thread to another. We address the problem of predicting if the course instructor would intervene on a thread, t. The instructor's decision to intervene, r, equals 0 when the instructor doesn't reply to the thread and 1 otherwise. The individual posts are not assumed to be labeled with any category and the only supervision given to the model during training is in form of intervention decision.

3.2 Logistic Regression (LR)

Our first attempt at solving this problem involved training a logistic regression for the binary prediction task which models $P(r|t)$.

3.2.1 Feature Engineering

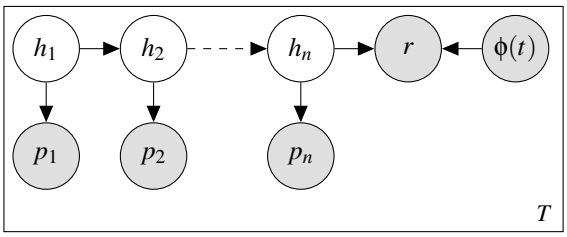
Our logistic regression model uses the following two types of features: *Thread only features* and *Aggregated post features*. 'Thread only features' capture information about the thread such as when, where, by who was the thread posted and lexical features based on the title of the thread. While these features provide a high-level information about the thread, it is also important to analyze the contents of the posts of the thread. In order to maintain a manageable feature space, we compress the features from posts and represent them using our 'Aggregated post features'.

Thread only features:

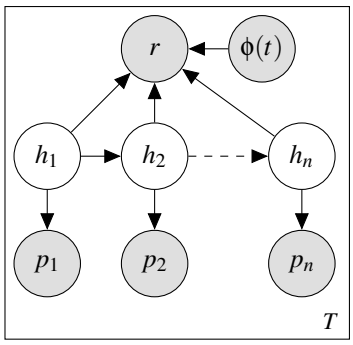
1. a binary feature indicating if the thread was started by an anonymous user
2. three binary features indicating whether the thread was marked as approved, unresolved or deleted (respectively)
3. forum id in which the thread was posted
4. time when the thread was started
5. time of last posting on the thread
6. total number of posts in the thread
7. a binary feature indicating if the thread title contains the words *lecture* or *lectures*
8. a binary feature indicating if the thread title contains the words *assignment*, *quiz*, *grade*, *project*, *exam* (and their plural forms)

Aggregated post features:

9. sum of number of votes received by the individual posts
10. mean and variance of the posting times of individual posts in the thread
11. mean of time difference between the posting times of individual posts and the closest course landmark. A course landmark is the deadline of an assignment, exam or project.
12. sum of count of occurrences of assessment related words e.g. *grade*, *exam*, *assignment*, *quiz*, *reading*, *project* etc. in the posts
13. sum of count of occurrences of words indicating technical problems e.g. *problem*, *error*
14. sum of count of occurrences of thread conclusive words like *thank you* and *thank*
15. sum of count of occurrences of *request*, *submit*, *suggest*



(a) Linear Chain Markov Model (LCMM)



(b) Global Chain Model (GCM)

Figure 2: Diagrams of the Linear Chain Markov Model (LCMM) and the Global Chain Model (GCM). p_i, r and $\phi(t)$ are observed and h_i are the latent variables. p_i and h_i represent the posts of the thread and their latent categories respectively; r represents the instructor's intervention and $\phi(t)$ represent the non-structural features used by the logistic regression model.

We had also considered and dropped (because of no performance gain) other features about identity of the user who started the thread, number of distinct participants in the thread (an important feature used by Backstrom et al. (2013)), binary feature indicating if the first and the last posts were by the same user, average number of words in the thread's posts, lexical features capturing references to the instructors in the posts etc.

3.3 Linear Chain Markov Model (LCMM)

The logistic regression model is good at exploiting the thread level features but not the content of individual posts. The 'Aggregated post features' attempt to capture this information but since the number of posts in a thread is variable, these features relied on aggregated values. We believe that considering aggregate values is not sufficient for the task in hand. As noted before, posts of a thread are not independent of each other. Instead, they are arranged chronologically such that a post is published in reply to the preceding posts and this might effect an instructor's decision to reply. For example, consider a thread that starts with a question. The following posts will be students' attempt to answer the question or raise further concerns or comment on previous posts. The instructor's post, though a future event, will be a part of this process.

For every thread, t, in the dataset:

1. Choose a start state, h_1, and emit the first post, p_1.
2. For every subsequent post, $p_i \ \forall \ i \ \in \{2 \dots n\}$:
   (a) Transition from h_{i-1} to h_i.
   (b) Emit post p_i.
3. Generate the instructor's intervention decision, r, using the last state h_n and non-structural features, $\phi(t)$.

Figure 3: Instructor's intervention decision process for the Linear Chain Markov Model.

We, therefore, propose to model this complete process using a linear chain markov model shown in Fig. 2a. The model abstractly represents the information from individual posts (p_i) using latent categories (h_i). The intervention decision, r, is the last step in the chain and thus incorporates information from the individual posts. It also depends on the thread level features: 'Thread only features' and the 'Aggregated post features' jointly represented by $\phi(t)$ (also referred to as the non-structural features). This process is explained in Fig. 3.

We use hand-crafted features to model the dynamics of the generative process. Whenever a latent state emits a post or transits to another latent state (or to the final intervention decision state), emission and transition features get fired which are then multiplied by respective weights to compute a thread's 'score':

$$f_w(t, p) = \max_h [\mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{p}, \boldsymbol{r}, \boldsymbol{h}, \boldsymbol{t})] \qquad (1)$$

Note that the non-structural features, $\phi(t)$, also contribute to the final score.

3.3.1 Learning and Inference

During training we maximize the combined scores of all threads in the dataset using a generic EM style algorithm. The supervision in this model is provided only in form of the observed intervention decision, r and the post categories, h_i are hid-

den. The model uses the pseudocode shown in Algorithm 1 to iteratively refine the weight vectors. In each iteration, the model first uses viterbi algorithm to decode thread sequences with the current weights w_t to find optimal highest scoring latent state sequences that agree with the observed intervention state ($r = r'$). In the next step, given the latent state assignments from the previous step, a structured perceptron algorithm (Collins, 2002) is used to update the weights w_{t+1} using weights from the previous step, w_t, initialization.

Algorithm 1 Training algorithm for LCMM

1: **Input:** Labeled data $D = \{(t, p, r)_i\}$
2: **Output:** Weights w
3: **Initialization:** Set w_j randomly, $\forall j$
4: **for** $t : 1$ to N **do**
5: $\hat{h_i} = \arg\max_h[\mathbf{w_t} \cdot \boldsymbol{\phi(p, r, h, t)}]$ such that $r = r_i \forall i$
6: w_{t+1} = StructuredPerceptron$(t, p, \hat{h}, r)$
7: **end for**
8: return w

While testing, we use the learned weights and viterbi decoding to compute the intervention state and the best scoring latent category sequence.

3.3.2 Feature Engineering

In addition to the 'Thread Only Features' and the 'Aggregated post features', $\phi(t)$ (Sec. 3.2.1, this model uses the following emission and transition features:

Post Emission Features:

1. $\phi(p_i, h_i)$ = count of occurrences of question words or question marks in p_i if the state is h_i; 0 otherwise.
2. $\phi(p_i, h_i)$ = count of occurrences of thank words (*thank you* or *thanks*) in p_i if the state is h_i; 0 otherwise.
3. $\phi(p_i, h_i)$ = count of occurrences of greeting words (e.g. *hi*, *hello*, *good morning*, *welcome* etc) in p_i if the state is h_i; 0 otherwise.
4. $\phi(p_i, h_i)$ = count of occurrences of assessment related words (e.g. *grade*, *exam*, *assignment*, *quiz*, *reading*, *project* etc.) in p_i if the state is h_i; 0 otherwise.
5. $\phi(p_i, h_i)$ = count of occurrences of *request*, *submit* or *suggest* in p_i if the state is h_i; 0 otherwise.
6. $\phi(p_i, h_i)$ = log(course duration/$t(p_i)$) if the state is h_i; 0 otherwise. Here $t(p_i)$ is the difference between the posting time of p_i and the closest course landmark (assignment or project deadline or exam).
7. $\phi(p_i, p_{i-1}, h_i)$ = difference between posting times of p_i and p_{i-1} normalized by course duration if the state is h_i; 0 otherwise.

Transition Features:

1. $\phi(h_{i-1}, h_i) = 1$ if previous state is h_{i-1} and current state is h_i; 0 otherwise.
2. $\phi(h_{i-1}, h_i, p_i, p_{i-1})$ = cosine similarity between p_{i-1} and p_i if previous state is h_{i-1} and current state is h_i; 0 otherwise.
3. $\phi(h_{i-1}, h_i, p_i, p_{i-1})$ = length of p_i if previous state is h_{i-1}, p_{i-1} has non-zero question words and current state is h_i; 0 otherwise.
4. $\phi(h_n, r) = 1$ if last post's state is h_n and intervention decision is r; 0 otherwise.
5. $\phi(h_n, r, p_n) = 1$ if last post's state is h_n, p_n has non-zero question words and intervention decision is r; 0 otherwise.
6. $\phi(h_n, r, p_n)$ = log(course duration/$t(p_n)$) if last post's state is h_n and intervention decision is r; 0 otherwise. Here $t(p_n)$ is the difference between the posting time of p_n and the closest course landmark (assignment or project deadline or exam).

3.4 Global Chain Model (GCM)

In this model we propose another way of incorporating the chain structure of a thread. Like the previous model, this model also assumes that posts belong to latent categories. It, however, doesn't model the instructor's intervention decision as a step in the thread generation process. Instead, it assumes that instructor's decision to intervene is dependent on all the posts in the threads, modeled using the latent post categories. This model is shown in Fig. 2b. Assuming that p represents posts of thread t, h represents the latent category assignments, r represents the intervention decision; feature vector, $\boldsymbol{\phi(p, r, h, t)}$, is extracted for each thread and using the weight vector, $\mathbf{w}$, this model defines a decision function, similar to what is shown in Equation 1.

3.4.1 Learning and Inference

Similar to the traditional maximum margin based Support Vector Machine (SVM) formulation, our model's objective function is defined as:

$$\min_w \frac{\lambda}{2}||\mathbf{w}||^2 + \sum_j^T l(-r_j f_w(t_j, p_j)) \quad (2)$$

where λ is the regularization coefficient, t_j is the j^{th} thread with intervention decision r_j and p_j are the posts of this thread. $\mathbf{w}$ is the weight vector, $l(\cdot)$ is the squared hinge loss function and $f_w(t_j, p_j)$ is defined in Equation 1.

Replacing the term $f_w(t_j, p_j)$ with the contents of Equation 1 in the minimization objective above, reveals the key difference from the traditional SVM formulation - the objective function has a maximum term inside the global minimization problem making it non-convex.

We, therefore, employ the optimization algorithm presented in (Chang et al., 2010) to solve this problem. Exploiting the semi-convexity property (Felzenszwalb et al., 2010), the algorithm works in two steps, each executed iteratively. In the first step, it determines the latent variable assignments for positive examples. The algorithm then performs two step iteratively - first it determines the structural assignments for the negative examples, and then optimizes the fixed objective function using a cutting plane algorithm. Once this process converges for negative examples, the algorithm reassigns values to the latent variables for positive examples, and proceeds to the second step. The algorithm stops once a local minimum is reached. A somewhat similar approach, which uses the Convex-Concave Procedure (CCCP) is presented by (Yu and Joachims, 2009).

At test time, given a thread, t, and it posts, p, we use the learned weights to compute $f_w(t, p)$ and classify it as belonging to the positive class (instructor intervenes) if $f_w(t, p) \geq 0$.

3.4.2 Feature Engineering

The feature set used by this model is very similar to the features used by the previous model. In addition to the non-structural features used by the logistic regression model (Sec. 3.2.1), it uses all the Post Emission features and the three transition features represented by $\phi(h_{i-1}, h_i)$ and $\phi(h_{i-1}, h_i, p_i, p_{i-1})$ as described in Sec. 3.3.2.

4 Empirical Evaluation

This section describes our experiments.

4.1 Datasets and Evaluation Measure

For our experiments, we have used the forum content of two MOOCs from different domains (science and humanities), offered by *Coursera*[1], a leading education technology company. Both courses were taught by professors from the University of Maryland, College Park.

Genes and the Human Condition (From Behavior to Biotechnology) (GHC) dataset. [2] This course was attended by 30,000 students and the instructional staff comprised of 2 instructors, 3 Teaching Assistants and 56 technical support staff. The discussion forum of this course consisted of 980 threads composed of about 3,800 posts.

Women and the Civil Rights Movement (WCR) dataset. [3] The course consisted of a classroom of about 14,600 students, 1 instructor, 6 Teaching Assistants and 49 support staff. Its discussion forum consisted of 800 threads and 3,900 posts.

We evaluate our models on held-out test sets. For the GHC dataset, the test set consisted of 186 threads out of which the instructor intervened on 24 while, for the WCR dataset, the instructor intervened on 21 out of 155 threads.

Also, it was commonly observed that after an instructor intervenes on a thread, its posting and/or viewing behavior increases. We, therefore, only consider the student posts until the instructor's first intervention. Care was also taken to not use features that increased/decreased disproportionately because of the instructor's intervention such as number of views or votes of a thread.

In our evaluation we approximate instructor's 'should reply' instances with those where the instructor indeed replied. Unlike general forum users, we believe that the correlation between the two scenarios is quite high for instructors. It is their responsibility to reply, and by choosing to a MOOC, they have 'bought in' to the idea of forum participation. The relatively smaller class sizes of these two MOOCs also ensured that most threads were manually reviewed, thus reducing instances of 'missed' threads while retaining the posting behavior and content of a typical MOOC.

4.2 Experimental Results

Since the purpose of solving this problem is to identify the threads which should be brought to the notice of the instructors, we measure the performance of our models using F-measure of the positive class. The values of various parameters were selected using 10-fold Cross Validation on

[1] https://www.coursera.org/

[2] https://www.coursera.org/course/genes

[3] https://www.coursera.org/course/womencivilrights

Model	Genes and the Human Condition (GHC)			Women and the Civil Rights (WCR)		
	P	R	F	P	R	F
LR	44.44	16.67	24.24	66.67	15.38	25.00
J48	45.50	20.80	28.55	25.00	23.10	24.01
LCMM	33.33	29.17	31.11	42.86	23.08	**30.00**
GCM	60.00	25.00	**35.29**	50.00	18.52	27.03

Table 1: Held-out test set performances of chain models, LCMM and GCM, are better than that of the unstructured models, LR and J48.

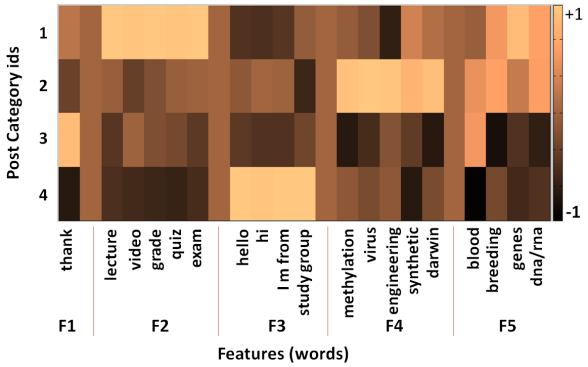


Figure 4: Visualization of lexical contents of the categories learnt by our model from the GHC dataset. Each row is a category and each column represents a feature vector. Bright cream color represents high values while lower values are represented by darker shades. Dark beige columns are used to better separate the five feature clusters, F1-F5, which represent words that are common in thanking, logistics-related, introductory, syllabus related and miscellaneous posts respectively. Categories 1,2,3 and 4 are dominated by F2, F4, F1 and F3 respectively indicating a semantic segregation of posts by our model's categories.

the training set. Table 1 presents the performances of the proposed models on the held-out test sets. We also report performance of a decision tree (J48) on the test sets for sake of comparison.

We can see that the chain based models, Linear Chain Markov Model (LCMM) and Global Chain Model (GCM), outperform the unstructured models, namely Logistic regression (LR) and Decision Trees (J48). This validates our hypothesis that using the post structure results in better modeling of instructor's intervention.

The table also reveals that GCM yields high precision and low recall values, which is possibly due to the model being more conservative owing to information from all posts of the thread.

4.3 Visual Exploration of Categories

Our chain based models assume that posts belong to different (latent) categories and use these categories to make intervention predictions. Since this process of discovering categories is data driven, it would be interesting to examine the contents of these categories. Fig. 4 presents a heat map of lexical content of categories identified by LCMM from the GHC dataset. The value of H (number of categories) was set to be 4 and was predetermined during the model selection procedure. Each row of the heat map represents a category and the columns represent values of individual features, $f(w, c)$, defined as: $f(w, c) = \frac{C(w,c)}{<C(w,c)>}$ where, $C(w, c)$ is total count of occurrences of a word, w, in all posts assigned to category, c and $< C(w, c) >$ represents its expected count based on its frequency in the dataset. While the actual size of vocabulary is huge, we use only a small subset of words in our feature vector for this visualization. These feature values, after normalization, are represented in the heat map using colors ranging from bright cream (high value) to dark black (low value). The darker the shade of a cell, the lower is the value represented by it.

For visual convenience, the features are manually clustered into five groups (F1 to F5) each separated by a dark beige colored column in the heat map. The first column of the heat map represents the F1 group which consists of words like *thank you, thanks* etc. These words are characteristic of posts that mark either the conclusion of a resolved thread or are posted towards the end of the course. Rows corresponding to the category 3 in Table 2 show two examples of such posts. Similarly, F2 represents the features related to logistics of the course and F3 captures introductory posts by new students. Finally, F4 contains words that are closely related to the subfield of gene and human conditions and would appear in posts that discuss specific aspects or chapters of the course con-

tents, while F5 contains general buzz words that would appear frequently in any biology course.

Analyzing individual rows of the heat map, we can see that out of F1 to F4, Categories 1, 2, 3 and 4 are dominated by logistics (F2), course content related (F4), thank you (F1) and introductory posts (F3) respectively, represented by bright colors in their respective rows. We also observe similar correlations while examining the columns of the heat map. Also, F5, which contains words common to the gene and human health domain, is scattered across multiple categories. For example, *dna/rna* and *breeding* are sufficiently frequent in category 1 as well as 2.

Table 2 gives examples of representative posts from the four clusters. Due to space constraints, we show only part of the complete post. We can see that these examples agree with our observations from the heat map.

Furthermore, as noted in Sec. 2, we compare the semantics of clusters learnt by our models with those proposed by Stump et al. (2013) even though the two categorizations are not directly comparable. Nevertheless, generally speaking, our category 1 corresponds to Stump et al. (2013)'s *Course structure/policies* and category 2 corresponds to *Content*. Interestingly, categories 3 and 4, which represent valedictory and introductory posts, correspond to a single *Social/affective* from the previous work.

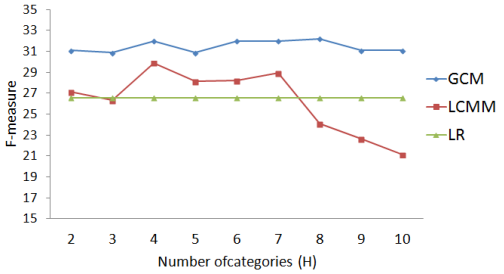
We can, therefore, conclude that the model, indeed splits the posts into categories that look semantically coherent to the human eyes.

4.4 Choice of Number of Categories

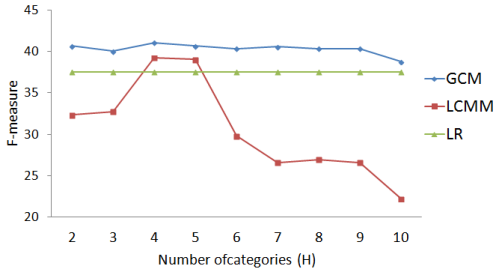
Our chain based models, assigning forum posts to latent categories, are parameterized with H, the number of categories. We therefore, study the sensitivity of our models to this parameter. Fig. 5, plots the 10-fold cross validation performance of the models with increasing values of H for the two datasets. Interestingly, the sensitivity of the two models to the value of H is very different.

The LCMM model's performance fluctuates as the value of H increases. The initial performance improvement might be due to an increase in the expressive power of the model. Performance peaks at $H = 4$ and then decreases, perhaps owing to over-fitting of the data.

In contrast, GCM performance remains steady for various values of H which might be attributed



(a) Genes and the Human Condition dataset



(b) Women and the Civil Rights Movement dataset

Figure 5: Cross validation performances of the two models with increasing number of categories.

to the explicit regularization coefficient which helps combat over-fitting, by encouraging zero weights for unnecessary categories.

4.5 How important are linguistic features?

We now focus on the structure independent features and experiment with their predictive value, according to types. We divide the features used by the LR into the following categories:[4]

- Full: set of all features (feature no. 1 to 15)
- lexical: based on content of thread titles and posts (feature no. 7 to 8 and 12 to 13)
- landmark: based on course landmarks (e.g, exams, quizzes) information (feature no. 11)
- MOOCs-specific: features specific to the MOOCs domain (lexical + landmark features)
- post: based only on aggregated posts information (feature no. 9 to 15)
- temporal: based on posting time patterns (feature no. 4, 5 and 10)

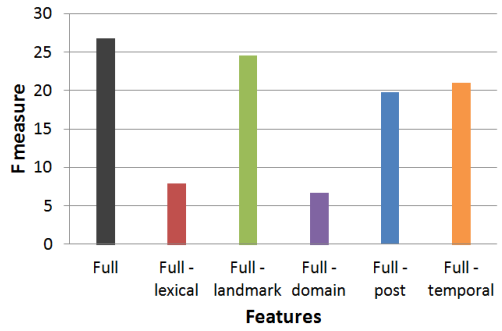
Fig. 6 shows 10-fold cross validation F-measure of the positive class for LR when different types of features are excluded from the full set.

The figure reveals that the MOOCs-specific features (purple bar) are important for both the datasets indicating a need for designing specialized models for forums analysis in this domain.

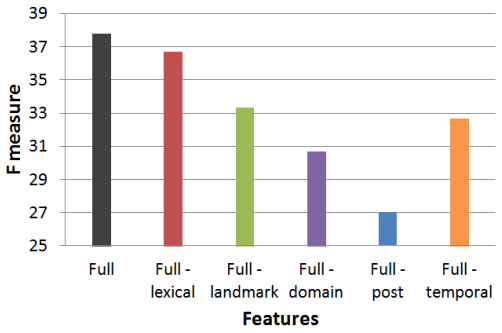
[4]Please refer to Sec 3.2.1 for description of the feature id.

Category	Example posts
1	'I'm having some issues with video playback. I have downloaded the videos to my laptop...'
1	'There was no mention of the nuclear envelope in the Week One lecture, yet it was in the quiz. Is this a mistake?'
2	'DNA methylation is a crucial part of normal development of organisms and cell differentiation in higher organisms...'
2	'In the lecture, she said there are...I don't see how tumor-suppressor genes are a cancer group mutation.'
3	'Thank you very much for a most enjoyable and informative course.'
3	'Great glossary! Thank you!'
4	'Hello everyone, I'm ... from the Netherlands. I'm a life science student.
4	'Hi, my name is ... this is my third class with coursera'

Table 2: Representative posts from the four categories learnt by our model. Due to space and privacy concerns we omit some parts of the text, indicated by ". . . ".



(a) Genes and the Human Condition dataset



(b) Women and the Civil Rights Movement dataset

Figure 6: Cross validation performances of the various feature types for the two datasets.

Also, lexical features (red bar) and post features (blue bar) have pretty dramatic effects in GHC and WCR data respectively.

Interestingly, removing the landmark feature set (green bar) causes a considerable drop in predictive performance, even though it consists of only one feature. Other temporal features (orange bar) also turn out to be important for the prediction. From a separate instructor activity vs time graph (not shown due to space constraints), we observed that instructors tend to get more active as the course progresses and their activity level also increases around quizzes/exams deadlines.

We can, therefore, conclude that all feature types are important and that lexical as well as MOOC specific analysis is necessary for modeling instructor's intervention.

5 Conclusion

One of the main challenges in MOOCs is managing student-instructor interaction. The massive scale of these courses rules out any form of personalized interaction, leaving instructors with the need to go over the forum discussions, gauge student reactions and selectively respond when appropriate. This time consuming and error prone task stresses the need for methods and tools supplying this actionable information automatically.

This paper takes a first step in that direction, and formulates the novel problem of predicting instructor intervention in MOOC discussion forums. Our main technical contribution is to construct predictive models combining information about forum post content and posting behavior with information about the course and its landmarks.

We propose three models for addressing the task. The first, a logistic regression model is trained on thread level and aggregated post features. The other two models take thread structure into account when making the prediction. These models assume that posts can be represented by categories which characterize post content at an abstract level, and treat category assignments as latent variables organized according to, and influenced by, the forum thread structure.

Our experiments on forum data from two different *Coursera* MOOCs show that utilizing thread structure is important for predicting instructor's behavior. Furthermore, our qualitative analysis shows that our latent categories are semantically coherent to human eye.

References

Ashton Anderson, Daniel P. Huttenlocher, Jon M. Kleinberg, and Jure Leskovec. 2014. Engaging with massive online courses. In *WWW*, pages 687–698.

Yoav Artzi, Patrick Pantel, and Michael Gamon. 2012. Predicting responses to microblog posts. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 602–606, Stroudsburg, PA, USA. Association for Computational Linguistics.

Erik Aumayr, Jeffrey Chan, and Conor Hayes. 2011. Reconstruction of threaded conversations in online discussion forums. In Lada A. Adamic, Ricardo A. Baeza-Yates, and Scott Counts, editors, *ICWSM*. The AAAI Press.

Lars Backstrom, Jon Kleinberg, Lillian Lee, and Cristian Danescu-Niculescu-Mizil. 2013. Characterizing and curating conversation threads: Expansion, focus, volume, re-entry. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 13–22, New York, NY, USA. ACM.

Eytan Bakshy, Jake M. Hofman, Winter A. Mason, and Duncan J. Watts. 2011. Everyone's an influencer: Quantifying influence on twitter. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 65–74, New York, NY, USA. ACM.

Prakhar Biyani, Cornelia Caragea, and Prasenjit Mitra. 2013. Predicting subjectivity orientation of online forum threads. In *CICLing (2)*, pages 109–120.

Rose Catherine, Rashmi Gangadharaiah, Karthik Visweswariah, and Dinesh Raghu. 2013. Semi-supervised answer extraction from discussion forums. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1–9, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Ming-Wei Chang, Dan Goldwasser, Dan Roth, and Vivek Srikumar. 2010. Discriminative learning over constrained latent representations. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 429–437, Stroudsburg, PA, USA. Association for Computational Linguistics.

Snigdha Chaturvedi, Vittorio Castelli, Radu Florian, Ramesh M. Nallapati, and Hema Raghavan. 2014. Joint question clustering and relevance prediction for open domain non-factoid question answering. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 503–514, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Munmun De Choudhury, Hari Sundaram, Ajita John, and Dorée Duncan Seligmann. 2009. What makes conversations interesting? themes, participants and consequences of conversations in online social media. In *18th International World Wide Web Conference (WWW)*, pages 331–331, April.

Mark Edmundson. 2012. The trouble with online education, July 19. `http://www.nytimes.com/2012/07/20/opinion/the-trouble-with-online-education.html`.

Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. 2010. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645.

Blaz Fortuna, Eduarda Mendes Rodrigues, and Natasa Milic-Frayling. 2007. Improving the classification of newsgroup messages through social network analysis. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 877–880, New York, NY, USA. ACM.

Dan Goldwasser and Hal Daumé III. 2014. "I object!" modeling latent pragmatic effects in courtroom dialogues. *European Chapter of the Association for Computational Linguistics (EACL)*, April. To appear.

Benjamin Golub and Matthew O. Jackson. 2010. Seeing only the successes: The power of selection bias in explaining the structure of observed internet diffusions.

Vicenç Gómez, Andreas Kaltenbrunner, and Vicente López. 2008. Statistical analysis of the social network and discussion threads in slashdot. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 645–654, New York, NY, USA. ACM.

Jizhou Huang, Ming Zhou, and Dan Yang. 2007. Extracting chatbox knowledge from online discussion forums. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 423–428, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Jonathan Huang, Chris Piech, Andy Nguyen, and Leonidas J. Guibas. 2013. Syntactic and functional variability of a million code submissions in a machine learning mooc. In *AIED Workshops*.

Minwoo Jeong, Chin-Yew Lin, and Gary Geunbae Lee. 2009. Semi-supervised speech act recognition in emails and forums. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, EMNLP '09, pages 1250–1259, Stroudsburg, PA, USA. Association for Computational Linguistics.

René F. Kizilcec, Chris Piech, and Emily Schneider. 2013. Deconstructing disengagement: analyzing learner subpopulations in massive open online courses. In *LAK*, pages 170–179.

Jon M. Kleinberg. 2013. Computational perspectives on social phenomena at global scales. In Francesca Rossi, editor, *IJCAI*. IJCAI/AAAI.

Ravi Kumar, Mohammad Mahdian, and Mary McGlohon. 2010. Dynamics of conversations. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 553–562, New York, NY, USA. ACM.

Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. 2010. What is twitter, a social network or a news media? In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 591–600, New York, NY, USA. ACM.

K. Lerman and R. Ghosh. 2010. Information contagion: An empirical study of the spread of news on digg and twitter social networks. In *Proceedings of 4th International Conference on Weblogs and Social Media (ICWSM)*.

David Liben-Nowell and Jon Kleinberg. 2008. Tracing the flow of information on a global scale using Internet chain-letter data. *Proceedings of the National Academy of Sciences*, 105(12):4633–4638, 25 March.

Andy Nguyen, Christopher Piech, Jonathan Huang, and Leonidas J. Guibas. 2014. Codewebs: scalable homework search for massive open online programming courses. In *WWW*, pages 491–502.

Chris Piech, Mehran Sahami, Daphne Koller, Steve Cooper, and Paulo Blikstein. 2012. Modeling how students learn to program. In *SIGCSE*, pages 153–160.

Chris Piech, Jonathan Huang, Zhenghao Chen, Chuong Do, Andrew Ng, and Daphne Koller. 2013. Tuned models of peer assessment in MOOCs. In *Proceedings of The 6th International Conference on Educational Data Mining (EDM 2013)*.

Arti Ramesh, Dan Goldwasser, Bert Huang, Hal Daumé III, and Lise Getoor. 2013. Modeling learner engagement in moocs using probabilistic soft logic. In *NIPS Workshop on Data Driven Education*.

Daniel M. Romero, Brendan Meeder, and Jon Kleinberg. 2011. Differences in the mechanics of information diffusion across topics: Idioms, political hashtags, and complex contagion on twitter. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 695–704, New York, NY, USA. ACM.

Glenda S. Stump, Jennifer DeBoer, Jonathan Whittinghill, and Lori Breslow. 2013. Development of a framework to classify mooc discussion forum posts: Methodology and challenges.

Manos Tsagkias, Wouter Weerkamp, and Maarten de Rijke. 2009. Predicting the volume of comments on online news stories. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 1765–1768, New York, NY, USA. ACM.

Yi-Chia Wang, Mahesh Joshi, and Carolyn Penstein Ros. 2007. A feature based approach to leveraging context for classifying newsgroup style discussion segments. In John A. Carroll, Antal van den Bosch, and Annie Zaenen, editors, *ACL*. The Association for Computational Linguistics.

Hongning Wang, Chi Wang, ChengXiang Zhai, and Jiawei Han. 2011. Learning online discussion structures by conditional random fields. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 435–444, New York, NY, USA. ACM.

Chunyan Wang, Mao Ye, and Bernardo A. Huberman. 2012. From user comments to on-line conversations. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 244–252, New York, NY, USA. ACM.

Li Wang, Su Nam Kim, and Timothy Baldwin. 2013. The utility of discourse structure in forum thread retrieval. In *AIRS*, pages 284–295.

Tae Yano and Noah A. Smith. 2010. What's worthy of comment? content and comment volume in political blogs. In William W. Cohen and Samuel Gosling, editors, *ICWSM*. The AAAI Press.

Chun-Nam John Yu and Thorsten Joachims. 2009. Learning structural svms with latent variables. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1169–1176, New York, NY, USA. ACM.

A Joint Graph Model for Pinyin-to-Chinese Conversion with Typo Correction*

Zhongye Jia and Hai Zhao†
MOE-Microsoft Key Laboratory for Intelligent Computing and Intelligent Systems,
Center for Brain-Like Computing and Machine Intelligence
Department of Computer Science and Engineering, Shanghai Jiao Tong University
800 Dongchuan Road, Shanghai 200240, China
jia.zhongye@gmail.com, zhaohai@cs.sjtu.edu.cn

Abstract

It is very import for Chinese language processing with the aid of an efficient input method engine (IME), of which pinyin-to-Chinese (PTC) conversion is the core part. Meanwhile, though typos are inevitable during user pinyin inputting, existing IMEs paid little attention to such big inconvenience. In this paper, motivated by a key equivalence of two decoding algorithms, we propose a joint graph model to globally optimize PTC and typo correction for IME. The evaluation results show that the proposed method outperforms both existing academic and commercial IMEs.

1 Introduction

1.1 Chinese Input Method

The daily life of Chinese people heavily depends on Chinese input method engine (IME), no matter whether one is composing an E-mail, writing an article, or sending a text message. However, every Chinese word inputted into computer or cellphone cannot be typed through one-to-one mapping of key-to-letter inputting directly, but has to go through an IME as there are thousands of Chinese characters for inputting while only 26 letter keys are available in the keyboard. An IME is an essential software interface that maps Chinese characters into English letter combinations. An efficient IME will largely improve the user experience of Chinese information processing.

Nowadays most of Chinese IMEs are pinyin based. Pinyin is originally designed as the phonetic symbol of a Chinese character (based on the standard modern Chinese, mandarin) , using Latin letters as its syllable notation. For example, the pinyin of the Chinese character "爱"(love) is "ài". Most characters usually have unique pinyin representations, while a few Chinese characters may be pronounced in several different ways, so they may have multiple pinyin representations. The advantage of pinyin IME is that it only adopts the pronunciation perspective of Chinese characters so that it is simple and easy to learn. But there are only less than 500 pinyin syllables in standard modern Chinese, compared with over 6,000 commonly used Chinese characters, which leads to serious ambiguities for pinyin-to-character mapping. Modern pinyin IMEs mostly use a "*sentence-based*" decoding technique (Chen and Lee, 2000) to alleviate the ambiguities. "*Sentence based*" means that IME generates a sequence of Chinese characters upon a sequence of pinyin inputs with respect to certain statistical criteria.

1.2 Typos and Chinese Spell Checking

Written in Chinese characters but not alphabets, spell checking for Chinese language is quite different from the same task for other languages. Since Chinese characters are entered via IME, those user-made typos do not immediately lead to spelling errors. When a user types a wrong letter, IME will be very likely to fail to generate the expected Chinese character sequence. Normally, the user may immediately notice the inputting error and then make corrections, which usually means doing a bunch of extra operations like cursor

*This work was partially supported by the National Natural Science Foundation of China (Grant No.60903119, Grant No.61170114, and Grant No.61272248), the National Basic Research Program of China (Grant No.2013CB329401), the Science and Technology Commission of Shanghai Municipality (Grant No.13511500200), and the European Union Seventh Framework Program (Grant No.247619).

†Corresponding author

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1512–1523,
Baltimore, Maryland, USA, June 23-25 2014.

movement, deletion and re-typing. Thus there are two separated sub-tasks for Chinese spell checking: 1. *typo checking* for user typed pinyin sequences which should be a built-in module in IME, and 2. *spell checking* for Chinese texts in its narrow sense, which is typically a module of word processing applications (Yang et al., 2012b). These two terms are often confused especially in IME related works such as (Chen and Lee, 2000) and (Wu et al., 2009).

Pinyin typos have always been a serious problem for Chinese pinyin IMEs. The user may fail to input the completely right pinyin simply because he/she is a dialect speaker and does not know the exact pronunciation for the expected character. This may be a very common situation since there are about seven quite different dialects in Chinese, among which being spoken languages, six are far different from the standard modern Chinese, mandarin. With the boom of smart-phones, pinyin typos worsen due to the limited size of soft keyboard, and the lack of physical feedback on the touch screen. However, existing practical IMEs only provide small patches to deal with typos such as *Fuzzy Pinyin* (Wu and Chen, 2004) and other language specific errors (Zheng et al., 2011b).

Typo checking and correction has an important impact on IME performance. When IME fails to correct a typo and generate the expected sentence, the user will have to take much extra effort to move the cursor back to the mistyped letter and correct it, which leads to very poor user experience (Jia and Zhao, 2013).

2 Related Works

The very first approach for Chinese input with typo correction was made by (Chen and Lee, 2000), which was also the initial attempt of "sentence-based" IME. The idea of "statistical input method" was proposed by modeling PTC conversion as a hidden Markov model (HMM), and using Viterbi (Viterbi, 1967) algorithm to decode the sequence. They solved the typo correction problem by decomposing the conditional probability $P(H|P)$ of Chinese character sequence H given pinyin sequence P into a language model $P(w_i|w_{i-1})$ and a typing model $P(p_i|w_i)$. The typing model that was estimated on real user input data was for typo correction. However, real user input data can be very noisy and not very convenient to obtain. As we will propose a joint model in this paper, such an individual typing model is not necessarily built in our approach.

(Zheng et al., 2011a) developed an IME system with typo correction called CHIME using noisy channel error model and language-specific features. However their model depended on a very strong assumption that input pinyin sequence should have been segmented into pinyin words by the user. This assumption does not really hold in modern "sentence-based" IMEs. We release this assumption since our model solves segmentation, typo correction and PTC conversion jointly.

Besides the common HMM approach for PTC conversion, there are also various methods such as: support vector machine (Jiang et al., 2007), maximum entropy (ME) model (Wang et al., 2006), conditional random field (CRF) (Li et al., 2009) and statistical machine translation (SMT) (Yang et al., 2012a; Wang et al., 2013c; Zhang and Zhao, 2013), etc.

Spell checking or typo checking was first proposed for English (Peterson, 1980). (Mays et al., 1991) addressed that spell checking should be done within a context, i.e., a sentence or a long phrase with a certain meaning, instead of only in one word. A recent spell correction work is (Li et al., 2006), where a distributional similarity was introduced for spell correction of web queries.

Early attempts for Chinese spelling checking could date back to (Chang, 1994) where character tables for similar shape, pronunciation, meaning, and input-method-code characters were proposed. More recently, the 7th SIGHAN Workshop on Chinese Language Processing (Yu et al., 2013) held a shared task on Chinese spell checking. Various approaches were made for the task including language model (LM) based methods (Chen et al., 2013), ME model (Han and Chang, 2013), CRF (Wang et al., 2013d; Wang et al., 2013a), SMT (Chiu et al., 2013; Liu et al., 2013), and graph model (Jia et al., 2013), etc.

3 Pinyin Input Method Model

3.1 From English Letter to Chinese Sentence

It is a rather long journey from the first English letter typed on the keyboard to finally a completed Chinese sentence generated by IME. We will first take an overview of the entire process.

The average length of pinyin syllables is about 3 letters. There are about 410 pinyin syllables used in the current pinyin system. Each pinyin sylla-

ble has a bunch of corresponding Chinese characters which share the same pronunciation represented by the syllable. The number of those homophones ranges from 1 to over 300. Chinese characters then form words. But word in Chinese is a rather vague concept. Without word delimiters, linguists have argued on what a Chinese word really is for a long time and that is why there is always a primary word segmentation treatment in most Chinese language processing tasks (Zhao et al., 2006; Huang and Zhao, 2007; Zhao and Kit, 2008; Zhao et al., 2010; Zhao and Kit, 2011; Zhao et al., 2013). A Chinese word may contain from 1 to over 10 characters due to different word segmentation conventions. Figure 1 demonstrates the relationship of pinyin and word, from pinyin letters "*nihao*" to the word "你好 *(hello)*". Typically, an IME takes the pinyin input, segments it into syllables, looks up corresponding words in a dictionary and generates a sentence with the candidate words.

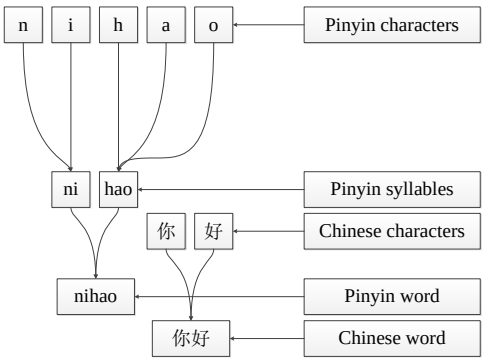


Figure 1: Relationship of pinyin and words

3.2 Pinyin Segmentation and Typo Correction

Non-Chinese users may feel confused or even surprised if they know that when typing pinyin through an IME, Chinese IME users will never enter delimiters such as "Space" key to segment either pinyin syllables or pinyin words, but just input the entire un-segmented pinyin sequence. For example, if one wants to input "你好世界 *(Hello world)*", he will just type "*nihaoshijie*" instead of segmented pinyin sequence "*ni hao shi jie*". Nevertheless, pinyin syllable segmentation is a much easier problem compared to Chinese word segmentation. Since pinyin syllables have a very limited vocabulary and follow a set of regularities strictly, it is convenient to perform pinyin syllable segmentation by using rules. But as the pinyin input is not segmented, it is nearly impossible to adopt previous spell checking methods for English to pinyin typo checking, although techniques for English spell checking have been well developed. A bit confusing but interesting, pinyin typo correction and segmentation come as two sides of one problem: when a pinyin sequence is mistyped, it is unlikely to be correctly segmented; when it is segmented in an awkward way, it is likely to be mistyped.

Inspired by (Yang et al., 2012b) and (Jia et al., 2013), we adopt the graph model for Chinese spell checking for pinyin segmentation and typo correction, which is based on the shortest path word segmentation algorithm (Casey and Lecolinet, 1996). The model has two major steps: segmentation and correction.

3.2.1 Pinyin Segmentation

The shortest path segmentation algorithm is based on the idea that a reasonable segmentation should minimize the number of segmented units. For a pinyin sequence $p_1 p_2 \dots p_L$, where p_i is a letter, first a directed acyclic graph (DAG) $G_S = (\mathbb{V}, \mathbb{E})$ is built for pinyin segmentation step. The vertex set $\mathbb{V}$ consists of the following parts:

- Virtual start vertex S_0 and end vertex S_E;
- Possible legal syllables fetched from dictionary $\mathbb{D}_p$ according to the input pinyin sequence:

$$\{S_{i,j} | S_{i,j} = p_i \dots p_j \in \mathbb{D}_p\};$$

- The letter itself as a fallback no matter if it is a legal pinyin syllable or not:

$$\{S_i | S_i = p_i\}.$$

The vertex weights w_S are all set to 0. The edges are from a syllable to all syllables next to it:

$$\mathbb{E} = \{E(S_{i,j} \to S_{j+1,k}) | S_{i,j}, S_{j+1,k} \in \mathbb{V}\}.$$

The edge weight the negative logarithm of conditional probability $P(S_{j+1,k}|S_{i,j})$ that a syllable $S_{i,j}$ is followed by $S_{j+1,k}$, which is give by a bigram language model of pinyin syllables:

$$W_{E(S_{i,j} \to S_{j+1,k})} = -\log P(S_{j+1,k}|S_{i,j})$$

The shortest path P^* on the graph is the path P with the least sum of weights:

$$P^* = \underset{(v,E) \in G \wedge (v,E) \in P}{\arg\min} \sum_v w_v + \sum_E W_E.$$

Computing the shortest path from S_0 to S_E on G_S yields the best segmentation. This is the single source shortest path (SSSP) problem on DAG which has an efficient algorithm by preprocessing the DAG with topology sort, then traversing vertices and edges in topological order. It has the time complexity of $O(|\mathbb{V}| + |\mathbb{E}|)$. For example, one intends to input "你好世界 *(Hello world)*" by typing "*nihaoshijie*", but mistyped as "*mihaoshijiw*". The graph for this input is shown in Figure 2. The shortest path, i.e., the best segmentation is "*mi hao shi ji w*". We will continue to use this example in the rest of this paper.

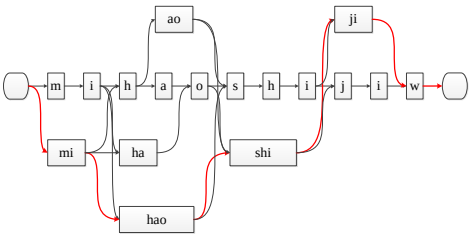


Figure 2: Graph model for pinyin segmentation

3.2.2 Pinyin Typo Correction

Next in the correction step, for the segmented pinyin sequence $S_1, S_2, \ldots, S_M$, a graph G_c is constructed to perform typo correction. The vertex set $\mathbb{V}$ consists of the following parts:

- Virtual start vertex S'_0 and end vertex S'_E with vertex weights of 0;
- All possible syllables similar to original syllable in G_s. If the adjacent syllables can be merged into a legal syllable, the merged syllable is also added into $\mathbb{V}$:

$$\{S'_{i,j} | S'_{i,j} = S'_i \ldots S'_j \in \mathbb{D}_p, \\ S'_k \sim S_k, k = i \le j\},$$

where the similarity $\sim$ is measured in Levenshtein distance (Levenshtein, 1966). Syllables with Levenshtein distance under a certain threshold are considered as similar:

$$\mathcal{L}(S_i, S_j) < T \leftrightarrow S_i \sim S_j.$$

The vertex weight is the Levenshtein distance multiply by a normalization parameter:

$$w_{S'_{i,j}} = \beta \sum_{k-i}^{j} \mathcal{L}(S'_k, S_k).$$

Similar to G_s, the edges are from one syllable to all syllables next to it and edge weights are the conditional probabilities between them. Computing the shortest path from S'_0 to S'_E on G_c yields the best typo correction result. In addition, the result has been segmented so far. Considering our running example, the graph G_c is shown in Figure 3, and the typo correction result is "*mi hao shi jie*".

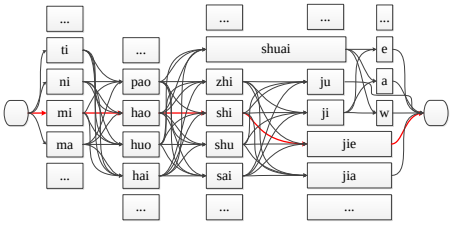


Figure 3: Graph model for pinyin typo correction

Merely using the above model, the typo correction result is not satisfying yet, no matter how much effort is paid. The major reason is that the basic semantic unit of Chinese language is actually word (tough vaguely defined) which is usually composed of several characters. Thus the conditional probability between characters does not make much sense. In addition, a pinyin syllable usually maps to dozens or even hundreds of corresponding homophonic characters, which makes the conditional probability between syllables much more noisy. However, using pinyin words instead of syllables is not a wise choice because pinyin word segmentation is not so easy a task as syllable segmentation. To make typo correction better, we consider to integrate it with PTC conversion using a joint model.

3.3 Hidden Markov Model for Pinyin-to-Chinese Conversion

PTC conversion has long been viewed as a decoding problem using HMM. We continue to follow this formalization. The best Chinese character sequence W^* for a given pinyin syllable sequence S is the one with the highest conditional probability $P(W|S)$ that

$$\begin{aligned} W^* &= \arg\max_W P(W|S) \\ &= \arg\max_W \frac{P(W)P(S|W)}{P(S)} \\ &= \arg\max_W P(W)P(S|W) \\ &= \arg\max_{w_1, w_w, \ldots, w_M} \prod_{w_i} P(w_i|w_{i-1}) \prod_{w_i} P(s_i|w_i) \end{aligned}$$

In the HMM for pinyin IME, observation states are pinyin syllables, hidden states are Chinese words, emission probability is $P(s_i|w_i)$, and transition probability is $P(w_i|w_{i-1})$. Note the transition probability is the conditional probability between words instead of characters. PTC conversion is to decode the Chinese word sequence from the pinyin sequence. The Viterbi algorithm (Viterbi, 1967) is used for the decoding.

The shortest path algorithm for typo correction and Viterbi algorithm for PTC conversion are very closely related. It has been strictly proven by (Forney, 1973) that the sequence decoding problem on HMM is formally identical to finding a shortest path on a certain graph, which can be constructed in the following manner.

Consider a first order HMM with all possible observations $\mathbb{O} = \{o_1, o_2, \ldots, o_M\}$, hidden states $\mathbb{S} = \{s_1, s_2, \ldots, s_N\}$, a special start state s_0, emission probabilities $(\mathcal{E}_{s_i,o_k}) = P(o_k|s_i)$, transition probabilities $(\mathcal{T}_{s_i,s_j}) = P(s_j|s_i)$, and start probabilities $(\mathcal{S}_{s_i}) = P(s_i|s_0)$. For an observation sequence of T time periods $Y = \{y_1, y_2, \ldots, y_T | y_t \in \mathbb{O}, t = 1, \ldots, T\}$, the decoding problem is to find the best corresponding hidden state sequence X^* with the highest probability, i.e.,

$$X^* = \arg\max_{x_1, x_t \in \mathbb{S}} \mathcal{S}_{x_1} \mathcal{E}_{x_1,y_1} \prod_{t=2}^{T} \mathcal{E}_{x_t,y_t} \mathcal{T}_{x_{t-1},x_t}. \quad (1)$$

Then we will construct a DAG $G = (\mathbb{V}, \mathbb{E})$ upon the HMM. The vertex set $\mathbb{V}$ includes:

- Virtual start vertex v_0 and end vertex v_E with vertex weight of 0;
- Normal vertices v_{x_t}, where $t = 1, \ldots, T$, and $\forall x_t \in \mathbb{S}$. The vertex weight is the negative logarithm of emission probability:

$$w_{v_{x_t}} = -\log \mathcal{E}_{x_t,y_t}.$$

The edge set $\mathbb{E}$ includes:

- Edges from the start vertex $E(v_0 \to v_{x_1})$ with edge weight

$$W_{E(v_0 \to v_{x_1})} = -\log \mathcal{S}_{x_1},$$

where $\forall x_1 \in \mathbb{S}$;

- Edges to the end vertex $E(v_{x_T} \to v_E)$ with vertex weights of 0;
- Edges between adjacent time periods $E(v_{x_{t-1}} \to v_{x_t})$ with edge weight

$$W_{E(v_{x_{t-1}} \to v_{x_t})} = -\log \mathcal{T}_{x_{t-1},x_t},$$

where $t = 2, \ldots, T$, and $\forall x_t, x_{t-1} \in \mathbb{S}$.

The shortest path P^* from v_0 to v_E is the one with the least sum of vertex and edge weights, i.e.,

$$\begin{aligned} P^* &= \arg\min_{v_{x_t} \in \mathbb{V}} \sum_{t=1}^{T} (w_{v_{x_t}} + W_{E(v_{x_{t-1}} \to v_{x_t})}) \\ &= \arg\min_{v_{x_1}, v_{x_t} \in \mathbb{V}} \{-\log \mathcal{S}_{x_1} - \log \mathcal{E}_{x_1,y_1} \\ &\quad + \sum_{t=2}^{T} (-\log \mathcal{E}_{x_t,y_t} - \log \mathcal{T}_{x_{t-1},x_t})\} \\ &= \arg\max_{v_{x_1}, v_{x_t} \in \mathbb{V}} \mathcal{S}_{x_1} \mathcal{E}_{x_1,y_1} \prod_{t=2}^{T} \mathcal{E}_{x_t,y_t} \mathcal{T}_{x_{t-1},x_t}. \quad (2) \end{aligned}$$

The optimization goal of P^* in Equation (2) is identical to that of X^* in Equation (1).

3.4 Joint Graph Model For Pinyin IME

Given HMM decoding problem is identical to SSSP problem on DAG, we propose a joint graph model for PTC conversion with typo correction. The joint graph model aims to find the global optimal for both PTC conversion and typo correction on the entire input pinyin sequence. The graph $G = (\mathbb{V}, \mathbb{E})$ is constructed based on graph G_c for typo correction in Section 3.2. The vertex set $\mathbb{V}$ consists of the following parts:

- Virtual start vertex V_0 and end vertex V_E with vertex weight of 0;
- Adjacent pinyin syllables in G_c are merged into pinyin words. Corresponding Chinese words are fetched from a PTC dictionary $\mathbb{D}_c$, which is a dictionary maps pinyin words to Chinese words, and added as vertices:

$$\{V_{i,j} | \forall V_{i,j} \in \mathbb{D}_c[S'_i \ldots S'_j], i \le j\};$$

The vertex weight consists of two parts: 1. the vertex weights of syllables in G_c, and 2. the emission probability:

$$\begin{aligned} w_{V_{i,j}} =& \beta \sum_{k=i}^{j} \mathcal{L}(S'_k, S_k) \\ & - \gamma \log P(S'_i \ldots S'_j | V_{i,j}); \end{aligned}$$

If the corresponding pinyin syllables in G_c have an edge between them, the vertices in G also have an edge:

$$\mathbb{E} = \{E(V_{i,j} \rightarrow V_{j+1,k}) | E(S_{i,j} \rightarrow S_{j+1,k}) \in \mathbb{G}_c\}.$$

The edge weights are the negative logarithm of the transition probabilities:

$$W_{E(V_{i,j} \rightarrow V_{j+1,k})} = -\log P(V_{j+1,k} | V_{i,j})$$

Although the model is formulated on first order HMM, i.e., the LM used for transition probability is a bigram one, it is easy to extend the model to take advantage of higher order n-gram LM, by tracking longer history while traversing the graph.

Computing the shortest path from V_0 to V_E on G yields the best pinyin-to-Chinese conversion with typo correction result. Considering our running example, the graph G is shown in Figure 4.

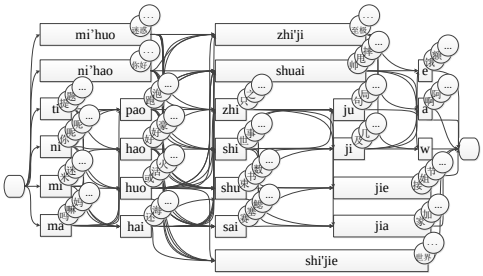


Figure 4: Joint graph model

The joint graph is rather huge and density. According to our empirical statistics, when setting threshold $T = 2$, for a sentence of M characters, the joint graph will have $|\mathbb{V}| = M \times 1,000$, and $|\mathbb{E}| = M \times 1,000,000$.

3.5 K-Shortest Paths

To reduce the scale of graph G, we filter graph G_c by searching its K-shortest paths first to get G'_c and construct G on top of G'_c. Figure 5 shows the 3-shortest paths filtered graph G'_c and Figure 6 shows the corresponding G for our running example. The scale of graph may be thus drastically reduced.

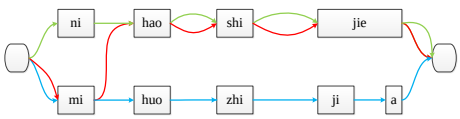


Figure 5: K-shortest paths in typo correction

An efficient heap data structure is required in K-shortest paths algorithm (Eppstein, 1998) for backtracking the best paths to current vertex while traversing. The heap is implemented as a priority queue of size K sorted according to path length that should support efficient push and pop operations. Fibonacci heap (Fredman and Tarjan, 1987) is adopted for the heap implementation since it has a push complexity of $O(1)$ which is better than the $O(K)$ for other heap structures.

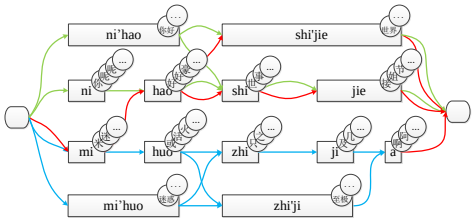


Figure 6: Filtered graph model

Another benefit provided by K-shortest paths is that it can be used for generating N-best candidates of PTC conversion, which may be helpful for further performance improvement.

4 Experiments

4.1 Corpora, Tools and Experiment Settings

The corpus for evaluation is the one provided in (Yang et al., 2012a), which is originally extracted from the *People's Daily* corpus and labeled with pinyin. The corpus has already been split into training **Train**, development **Dev** and test **Test** sets as shown in Table 1.

	Train	**Dev**	**Test**
#Sentence	1M	2K	100K
#character	43,679,593	83,765	4,123,184

Table 1: Data set size

SRILM (Stolcke, 2002) is adopted for language model training and KenLM (Heafield, 2011; Heafield et al., 2013) for language model query. The Chinese part of the corpus is segmented into words before LM training. Maximum matching word segmentation is used with a large word vocabulary $\mathcal{V}$ extracted from web data provided by (Wang et al., 2013b). The pinyin part is segmented according to the Chinese part. This vocabulary $\mathcal{V}$ also serves as the PTC dictionary. The original vocabulary is not labeled with pinyin, thus we use the PTC dictionary of *sunpinyin*[1] which is an open source Chinese pinyin IME, to label the

[1] http://code.google.com/p/sunpinyin/

vocabulary $\mathcal{V}$ with pinyin. The emission probabilities are estimated using the lexical translation module of MOSES (Koehn et al., 2007) as "translation probability" from pinyin to Chinese.

4.2 Evaluation Metrics

We will use conventional sequence labeling evaluation metrics such as sequence accuracy and character accuracy[2].

Chinese characters in a sentence may be separated by digits, punctuation and alphabets which are directly inputted without the IME. We follow the so-called term *Max Input Unit* (MIU), the longest consecutive Chinese character sequence proposed by (Jia and Zhao, 2013). We will mainly consider MIU accuracy (MIU-Acc) which is the ratio of the number of completely corrected generated MIUs over the number of all MIUs, and character accuracy (Ch-Acc), but the sentence accuracy (S-Acc) will also be reported in evaluation results.

We will also report the conversion error rate (ConvER) proposed by (Zheng et al., 2011a), which is the ratio of the number of mistyped pinyin word that is not converted to the right Chinese word over the total number of mistyped pinyin words[3].

4.3 Baseline System without Typo Correction

Firstly we build a baseline system without typo correction which is a pipeline of pinyin syllable segmentation and PTC conversion. The baseline system takes a pinyin input sequence, segments it into syllables, and then converts it to Chinese character sequence.

The pinyin syllable segmentation already has very high (over 98%) accuracy with a trigram LM using improved Kneser-Ney smoothing. According to our empirical observation, emission probabilities are mostly 1 since most Chinese words have unique pronunciation. So in this step we set $\gamma = 0$. We consider different LM smoothing methods including Kneser-Ney (KN), improved Kneser-Ney (IKN), and Witten-Bell (WB). All of the three smoothing methods for bigram and trigram LMs are examined both using back-off models and interpolated models. The number of N-best candidates for PTC conversion is set to 10. The results on **Dev** are shown in Figure 7 in which the "-i" suffix indicates using interpolated model. According to the results, we then choose the trigram LM using Kneser-Ney smoothing with interpolation.

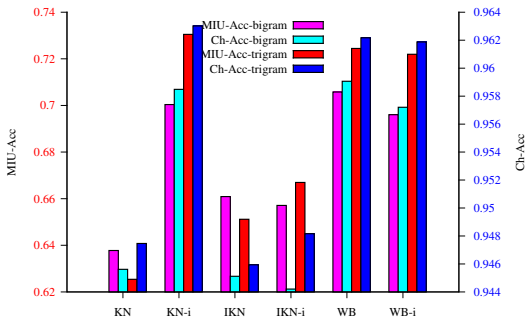


Figure 7: MIU-Acc and Ch-Acc with different LM smoothing

The choice of the number of N-best candidates for PTC conversion also has a strong impact on the results. Figure 8 shows the results on **Dev** with different Ns, of which the N axis is drawn in logarithmic scale. We can observe that MIU-Acc slightly decreases while N goes up, but Ch-Acc largely increases. We therefore choose $N = 10$ as trade-off.

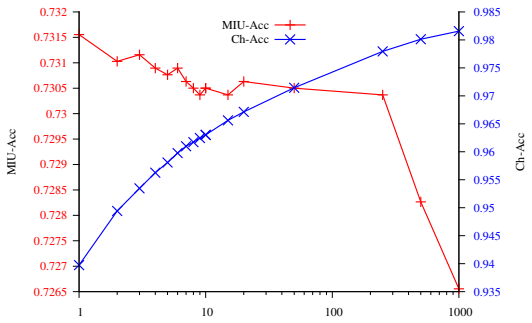


Figure 8: MIU-Acc and Ch-Acc with different Ns

The parameter γ determines emission probability. Results with different γ on **Dev** is shown in Figure 9, of which the γ axis is drawn in logarithmic scale. $\gamma = 0.03$ is chosen at last.

We compare our baseline system with several practical pinyin IMEs including *sunpinyin* and *Google Input Tools* (Online version)[4]. The results on **Dev** are shown in Table 2.

[2]We only work on the PTC conversion part of IME, thus we are unable to use existing evaluation systems (Jia and Zhao, 2013) for full Chinese IME functions.

[3]Other evaluation metrics are also proposed by (Zheng et al., 2011a) which is only suitable for their system since our system uses a joint model

[4]http://www.google.com/inputtools/try/

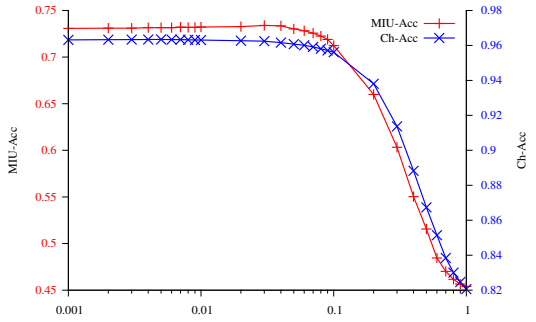


Figure 9: MIU-Acc and Ch-Acc with different γ

	MIU-Acc	Ch-Acc	S-Acc
Baseline	73.39	96.24	38.00
sunpinyin	52.37	87.51	13.95
Google	74.74	94.81	40.2
Yang-ME	-	93.3	30.2
Yang-MT	-	95.5	45.4

Table 2: Baseline system compared to other IMEs (%)

4.4 PTC Conversion with Typo Correction

Based upon the baseline system, we build the joint system of PTC conversion with typo correction.

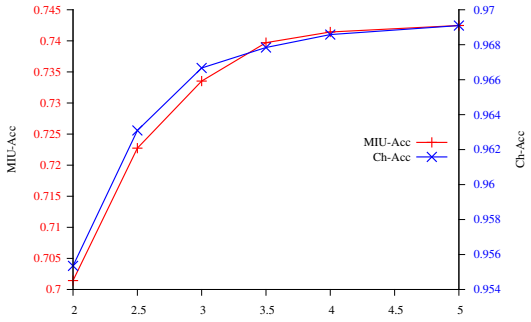
We simulate user typos by randomly generating errors automatically on the corpus. The typo rate is set according to previous Human-Computer Interaction (HCI) studies. Due to few works have been done on modeling Chinese text entry, we have to refer to those corresponding results on English (Wobbrock and Myers, 2006; MacKenzie and Soukoreff, 2002; Clarkson et al., 2005), which show that the average typo rate is about 2%. (Zheng et al., 2011a) performed an experiment that 2,000 sentences of 11,968 Chinese words were entered by 5 native speakers. The collected data consists of 775 mistyped pinyin words caused by one edit operation, and 85 caused by two edit operations. As we observe on **TRAIN** that the average pinyin word length is 5.24, then typo rate in the experiment of (Zheng et al., 2011a) can be roughly estimated as:

$$\frac{775 + 85 \times 2}{11968 \times 5.24} = 1.51\%,$$

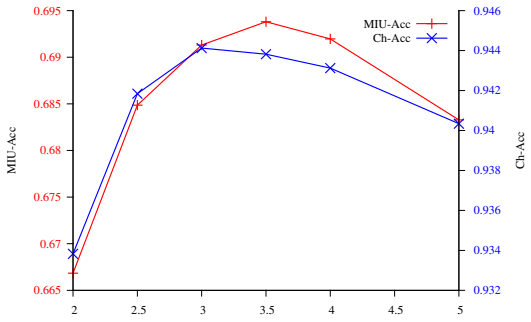
which is similar to the conclusion on English. Thus we generate corpora from **DEV** with typo rate of 0% (**0-P**), 2% (**2-P**), and 5% (**5-P**) to evaluate the system.

According to (Zheng et al., 2011a) most mistyped pinyin words are caused by one edit operation. Since pinyin syllable is much shorter than pinyin word, this ratio can be higher for pinyin syllables. From our statistics on **TRAIN**, with 2% randomly generated typos, $Pr(\mathcal{L}(S', S) < 2) = 99.86\%$. Thus we set the threshold T for $\mathcal{L}$ to 2.

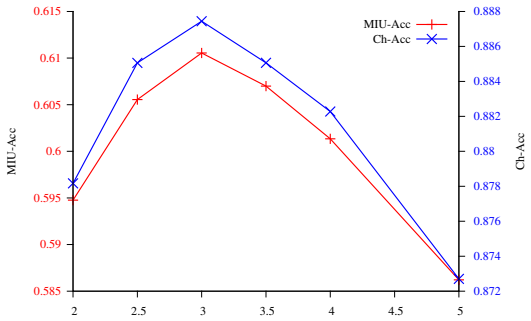
We first set K-shortest paths filter to $K = 10$ and tune β. Results with different β are shown in Figure 10. With $\beta = 3.5$, we select K. Results with different K are shown in Figure 11. We choose $K = 20$ since there is no significant improvement when $K > 20$.



(a) **0-P**



(b) **2-P**

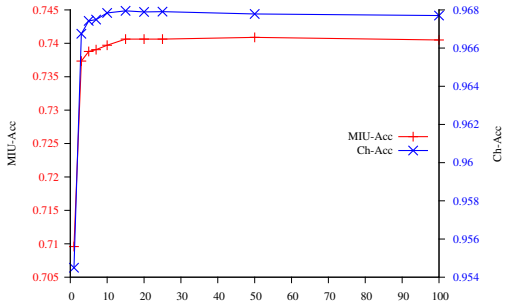


(c) **5-P**

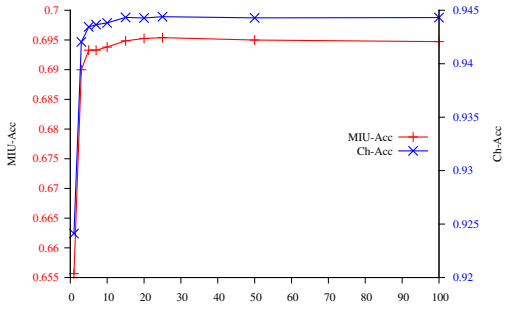
Figure 10: MIU-Acc and Ch-Acc with different β

The selection of K also directly guarantees the running time of the joint model. With $K = 20$, on a normal PC with Intel Pentium Dual-Core E6700 CPU, the PTC conversion rate is over 2000 characters-per-minute (*cpm*), which is much faster than the normal typing rate of 200 *cpm*.

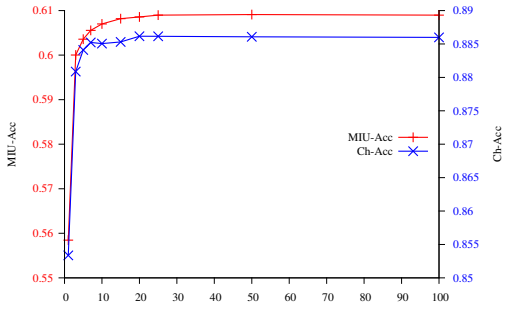
With all parameters optimized, results on **TEST**



(a) **0-P**



(b) **2-P**



(c) **5-P**

Figure 11: MIU-Acc and Ch-Acc with different K

using the proposed joint model are shown in Table 3 and Table 4. Our results are compared to the baseline system without typo correction and *Google Input Tool*. Since *sunpinyin* does not have typo correction module and performs much poorer than our baseline system, we do not include it in the comparison. Though no direct proofs can be found to indicate if Google Input Tool has an independent typo correction component, its outputs show that such a component is unlikely available.

Since *Google Input Tool* has to be accessed through a web interface and the network connection cannot be guaranteed. we only take a subset of 10K sentences of **TEST** to perform the experiments, and the results are shown in Table 3.

The scores reported in (Zheng et al., 2011a) are not listed in Table 4 since the data set is different. They reported a ConvER of 53.56%, which is given here for reference.

Additionally, to further inspect the robustness of our model, performance with typo rate ranges from 0% to 5% is shown in Figure 12. Although the performance decreases while typo rate goes up, it is still quite satisfying around typo rate of 2% which is assumed to be the real world situation.

	MIU-Acc	Ch-Acc	S-Acc	ConvER
Baseline **0-P**	79.90	97.47	48.87	-
Baseline **2-P**	50.47	90.53	11.12	99.95
Baseline **5-P**	30.26	82.83	3.32	99.99
Google **0-P**	79.08	95.26	46.83	-
Google **2-P**	49.47	61.50	11.08	91.70
Google **5-P**	29.18	36.20	3.29	94.64
Joint **0-P**	79.90	97.52	49.27	-
Joint **2-P**	75.55	95.40	40.69	18.45
Joint **5-P**	67.76	90.17	27.86	24.68

Table 3: Test results on 10K sentences from **TEST** (%)

	MIU-Acc	Ch-Acc	S-Acc	ConvER
Baseline **0-P**	74.46	96.42	40.50	-
Baseline **2-P**	47.25	89.50	9.62	99.95
Baseline **5-P**	28.28	81.74	2.63	99.98
Joint **2-P**	74.22	96.39	40.34	-
Joint **2-P**	69.91	94.14	33.11	21.35
Joint **5-P**	62.14	88.49	22.62	27.79

Table 4: Test results on **TEST** (%)

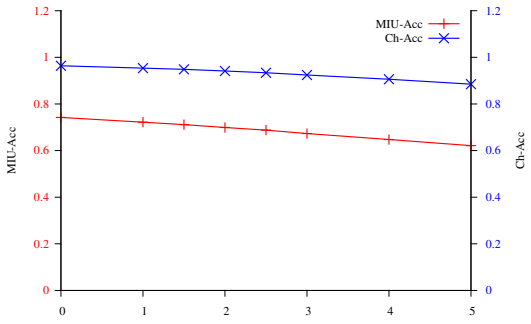


Figure 12: MIU-Acc and Ch-Acc with different typo rate (%)

5 Conclusion

In this paper, we have developed a joint graph model for pinyin-to-Chinese conversion with typo correction. This model finds a joint global optimal for typo correction and PTC conversion on the entire input pinyin sequence. The evaluation results show that our model outperforms both previous academic systems and existing commercial products. In addition, the joint model is efficient enough for practical use.

References

Richard G. Casey and Eric Lecolinet. 1996. A Survey of Methods and Strategies in Character Segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(7):690–706.

Chao-Huang Chang. 1994. A Pilot Study on Automatic Chinese Spelling Error Correction. *Journal of Chinese Language and Computing*, 4:143–149.

Zheng Chen and Kai-Fu Lee. 2000. A New Statistical Approach To Chinese Pinyin Input. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 241–247, Hong Kong, October.

Kuan-Yu Chen, Hung-Shin Lee, Chung-Han Lee, Hsin-Min Wang, and Hsin-Hsi Chen. 2013. A Study of Language Modeling for Chinese Spelling Check. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 79–83, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Hsun-wen Chiu, Jian-cheng Wu, and Jason S. Chang. 2013. Chinese Spelling Checker Based on Statistical Machine Translation. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 49–53, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Edward Clarkson, James Clawson, Kent Lyons, and Thad Starner. 2005. An Empirical Study of Typing Rates on mini-QWERTY Keyboards. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '05, pages 1288–1291, New York, NY, USA. ACM.

David Eppstein. 1998. Finding the K Shortest Paths. *SIAM Journal on computing*, 28(2):652–673.

Jr G. David Forney. 1973. The Viterbi Algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Michael L. Fredman and Robert Endre Tarjan. 1987. Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms. *Journal of the ACM (JACM)*, 34(3):596–615, July.

Dongxu Han and Baobao Chang. 2013. A Maximum Entropy Approach to Chinese Spelling Check. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 74–78, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable Modified Kneser-Ney Language Model Estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 690–696, Sofia, Bulgaria, August.

Kenneth Heafield. 2011. KenLM: Faster and Smaller Language Model Queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom, July.

Changning Huang and Hai Zhao. 2007. Chinese Word Segmentation: A Decade Review. *Journal of Chinese Information Processing*, 21(3):8–20.

Zhongye Jia and Hai Zhao. 2013. KySS 1.0: a Framework for Automatic Evaluation of Chinese Input Method Engines. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1195–1201, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Zhongye Jia, Peilu Wang, and Hai Zhao. 2013. Graph Model for Chinese Spell Checking. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 88–92, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Wei Jiang, Yi Guan, Xiaolong Wang, and BingQuan Liu. 2007. PinYin-to-Character Conversion Model based on Support Vector Machines. *Journal of Chinese information processing*, 21(2):100–105.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June. Association for Computational Linguistics.

Vladimir I. Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. In *Soviet physics doklady*, volume 10, page 707.

Mu Li, Muhua Zhu, Yang Zhang, and Ming Zhou. 2006. Exploring Distributional Similarity Based Models for Query Spelling Correction. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 1025–1032, Sydney, Australia, July. Association for Computational Linguistics.

Lu Li, Xuan Wang, Xiao-Long Wang, and Yan-Bing Yu. 2009. A Conditional Random Fields Approach to Chinese Pinyin-to-Character Conversion. *Journal of Communication and Computer*, 6(4):25–31.

Xiaodong Liu, Kevin Cheng, Yanyan Luo, Kevin Duh, and Yuji Matsumoto. 2013. A Hybrid Chinese Spelling Correction Using Language Model and Statistical Machine Translation with Reranking. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 54–58, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

I. Scott MacKenzie and R. William Soukoreff. 2002. A Character-level Error Analysis Technique for Evaluating Text Entry Methods. In *Proceedings of the Second Nordic Conference on Human-computer Interaction*, NordiCHI '02, pages 243–246, New York, NY, USA. ACM.

Eric Mays, Fred J Damerau, and Robert L Mercer. 1991. Context Based Spelling Correction. *Information Processing & Management*, 27(5):517–522.

James L. Peterson. 1980. Computer Programs for Detecting and Correcting Spelling Errors. *Commun. ACM*, 23(12):676–687, December.

Andreas Stolcke. 2002. SRILM-An Extensible Language Modeling Toolkit. In *Proceedings of the international conference on spoken language processing*, volume 2, pages 901–904.

Andrew J. Viterbi. 1967. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *Information Theory, IEEE Transactions on*, 13(2):260–269.

Xuan Wang, Lu Li, Lin Yao, and Waqas Anwar. 2006. A Maximum Entropy Approach to Chinese Pin Yin-To-Character Conversion. In *Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on*, volume 4, pages 2956–2959. IEEE.

Chun-Hung Wang, Jason S. Chang, and Jian-Cheng Wu. 2013a. Automatic Chinese Confusion Words Extraction Using Conditional Random Fields and the Web. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 64–68, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Peilu Wang, Ruihua Sun, Hai Zhao, and Kai Yu. 2013b. A New Word Language Model Evaluation Metric for Character Based Languages. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, pages 315–324. Springer.

Rui Wang, Masao Utiyama, Isao Goto, Eiichro Sumita, Hai Zhao, and Bao-Liang Lu. 2013c. Converting Continuous-Space Language Models into N-Gram Language Models for Statistical Machine Translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 845–850, Seattle, Washington, USA, October. Association for Computational Linguistics.

Yih-Ru Wang, Yuan-Fu Liao, Yeh-Kuang Wu, and Liang-Chun Chang. 2013d. Conditional Random Field-based Parser and Language Model for Traditional Chinese Spelling Checker. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 69–73, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Jacob O. Wobbrock and Brad A. Myers. 2006. Analyzing the Input Stream for Character- Level Errors in Unconstrained Text Entry Evaluations. *ACM Trans. Comput.-Hum. Interact.*, 13(4):458–489, December.

Jun Wu and Liren Chen. 2004. Fault-tolerant Romanized Input Method for Non-roman Characters, August 25. US Patent App. 10/928,131.

Jun Wu, Hulcan Zhu, and Hongjun Zhu. 2009. Systems and Methods for Translating Chinese Pinyin to Chinese Characters, January 13. US Patent 7,478,033.

Shaohua Yang, Hai Zhao, and Bao-liang Lu. 2012a. A Machine Translation Approach for Chinese Whole-Sentence Pinyin-to-Character Conversion. In *Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation*, pages 333–342, Bali,Indonesia, November. Faculty of Computer Science, Universitas Indonesia.

Shaohua Yang, Hai Zhao, Xiaolin Wang, and Bao-liang Lu. 2012b. Spell Checking for Chinese. In *International Conference on Language Resources and Evaluation*, pages 730–736, Istanbul, Turkey, May.

Liang-Chih Yu, Yuen-Hsien Tseng, Jingbo Zhu, and Fuji Ren, editors. 2013. *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*. Asian Federation of Natural Language Processing, Nagoya, Japan, October.

Jingyi Zhang and Hai Zhao. 2013. Improving Function Word Alignment with Frequency and Syntactic Information. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 2211–2217. AAAI Press.

Hai Zhao and Chunyu Kit. 2008. Exploiting Unlabeled Text with Different Unsupervised Segmentation Criteria for Chinese Word Segmentation. *Research in Computing Science*, 33:93–104.

Hai Zhao and Chunyu Kit. 2011. Integrating Unsupervised and Supervised Word Segmentation: The Role of Goodness Measures. *Information Sciences*, 181(1):163–183.

Hai Zhao, Chang-Ning Huang, and Mu Li. 2006. An Improved Chinese Word Segmentation System with Conditional Random Field. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 162–165, Sydney, Australia, July. Association for Computational Linguistics.

Hai Zhao, Chang-Ning Huang, Mu Li, and Bao-Liang Lu. 2010. A Unified Character-Based Tagging Framework for Chinese Word Segmentation. *ACM Transactions on Asian Language Information Processing (TALIP)*, 9(2):5.

Hai Zhao, Masao Utiyama, Eiichiro Sumita, and Bao-Liang Lu. 2013. An Empirical Study on Word Segmentation for Chinese Machine Translation. In *Computational Linguistics and Intelligent Text Processing*, pages 248–263. Springer.

Yabin Zheng, Chen Li, and Maosong Sun. 2011a. CHIME: An Efficient Error-tolerant Chinese Pinyin Input Method. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, pages 2551–2556. AAAI Press.

Yabin Zheng, Lixing Xie, Zhiyuan Liu, Maosong Sun, Yang Zhang, and Liyun Ru. 2011b. Why Press Backspace? Understanding User Input Behaviors in Chinese Pinyin Input Method. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Techologies*, pages 485–490, Portland, Oregon, USA, June. Association for Computational Linguistics.

Smart Selection

Patrick Pantel
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
ppantel@microsoft.com

Michael Gamon
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
mgamon@microsoft.com

Ariel Fuxman
Microsoft Research
1065 La Avenida St.
Mountain View, CA 94043, USA
arielf@microsoft.com

Abstract

Natural touch interfaces, common now in devices such as tablets and smartphones, make it cumbersome for users to select text. There is a need for a new text selection paradigm that goes beyond the high acuity selection-by-mouse that we have relied on for decades. In this paper, we introduce such a paradigm, called *Smart Selection*, which aims to recover a user's intended text selection from her touch input. We model the problem using an ensemble learning approach, which leverages multiple linguistic analysis techniques combined with information from a knowledge base and a Web graph. We collect a dataset of true intended user selections and simulated user touches via a large-scale crowdsourcing task, which we release to the academic community. We show that our model effectively addresses the smart selection task and significantly outperforms various baselines and standalone linguistic analysis techniques.

1 Introduction

The process of using a pointing device to select a span of text has a long history dating back to the invention of the mouse. It serves to access functions on text spans, such as copying/pasting, looking up a word in a dictionary, searching the Web, or accessing other accelerators. As consumers move from traditional PCs to mobile devices (e.g., tablets and smartphones), touch interaction is replacing the pointing devices of yore. Although more intuitive and arguably a more natural form of interaction, touch offers much less acuity (colloquially referred to as the *fat finger* problem). To select multi-word spans today, mobile devices require an intricate series of gestures that results in cumbersome user experiences[1]. Consequently, there is an opportunity to reinvent the way users select text in such devices.

Our task is, given a single user touch, to predict the span that the user likely intended to select. We call this task **smart selection**. We restrict our prediction task to cases where a user intends to perform research on a text span (dictionary/thesaurus lookup, translation, searching). We specifically consider operations on text spans that do not form a single unit (i.e., an entity, a concept, a topic, etc.) to be out of scope. For example, full sentences, paragraph and page fragments are out of scope.

Smart selection, as far as we know, is a new research problem. Yet there are many threads of research in the NLP community that identify multi-word sequences, which have coherent properties. For example, named-entity recognizers identify entities such as people/places/organizations, chunkers and parsers identify syntactic constituents such as noun phrases, key phrase detectors or term segmentors identify term boundaries. While each of these techniques retrieve meaningful linguistic units, our problem is a semantic one of recovering a user's *intent*, and as such none alone solves the entire smart selection problem.

In this paper, we model the problem of smart selection using an ensemble learning approach. We leverage various linguistic techniques, such as those discussed above, and augment them with other sources of information from a knowledge

[1] In order to select a multi-word span, a user would first have to touch on either word, then drag the left and right boundary handles to expand it to the adjacent words.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1524–1533,
Baltimore, Maryland, USA, June 23-25 2014.

base and a web graph. We evaluate our methods using a novel dataset constructed for our task. We construct our dataset of true user-intended selections by crowdsourcing the task of a user selecting spans of text in a researching task. We obtain 13,681 data points. For each intended selection, we construct test cases for each individual sub-word, simulating the user selecting via touch. The resulting testset consists of 33,912 $\langle$simulated selection, intended selection$\rangle$-pairs, which we further stratify into head, torso, and tail subsets. We release the full dataset and testset to the academic community for further research on this new NLP task. Finally, we empirically show that our ensemble model significantly improves upon various baseline systems.

In summary, the major contributions of our research are:

- We introduce a new natural language processing task, called *smart selection*, which aims to address an important problem in text selection for touch-enabled devices;
- We conduct a large crowd-sourced user study to collect a dataset of intended selections and simulated user selections, which we release to the academic community;
- We propose a machine-learned ensemble model for smart selection, which combines various linguistic annotation methods with information from a large knowledge base and web graph;
- We empirically show that our model can effectively address the smart selection task.

2 Related Work

Related work falls into three broad categories: linguistic unit detection, human computer interaction (HCI), and intent detection.

2.1 Linguistic Unit Detection

Smart selection is closely related to the detection of syntactic and semantic units: user selections are often entities, noun phrases, or concepts. A first approach to solving smart selection is to select an entity, noun phrase, or concept that subsumes the user selection. However, no single approach alone can cover the entire smart selection problem. For example, consider an approach that uses a state-of-the-art named-entity recognizer (NER) (Chinchor, 1998; Tjong Kim Sang and De Meulder, 2003; Finkel et al., 2005; Ratinov and Roth, 2009). We found in our dataset (see Section 3.2) that only a quarter of what users intend to select consists in fact of named entities. Although an NER approach can be very useful, it is certainly not sufficient. The remainder of the data can be partially addressed with noun phrase (NP) detectors (Abney, 1991; Ramshaw and Marcus, 1995; Muñoz et al., 1999; Kudo and Matsumoto, 2001) and lists of items in a knowledge base (KB), but again, each is not alone sufficient. NP detectors and KB-based methods are further very susceptible to the generation of false positives (i.e., text contains many nested noun phrases and knowledge base items include highly ambiguous terms).

In our work, we leverage all three techniques in order to benefit from their complementary coverage of user selections. We further create a novel unit detector, called the hyperlink intent model. Based on the assumption that Wikipedia anchor texts are similar in nature to what users would select in a researching task, it models the problem of recovering Wikipedia anchor texts from partial selections.

2.2 Human Computer Interaction

There is a substantial amount of research in the HCI community on how to facilitate interaction of a user with touch and speech enabled devices. To give but a few examples of trends in this field, Gunawardana et al. (2010) address the fat finger problem in the use of soft keyboards on mobile devices, Kumar et al. (2012) explore a novel speech interaction paradigm for text entry, and Sakamoto et al. (2013) introduce a technique that combines touch and voice input on a mobile device for improved navigation of user interface elements such as commands and controls. To the best of our knowledge, however, the problem of smart selection as we defined it has not been addressed.

2.3 Intent detection

There is a long line of research in the web literature on understanding user intent. The closest to smart selection is query recommendation (Baeza-Yates et al., 2005; Zhang and Nasraoui, 2006; Boldi et al., 2008), where the goal is to suggest queries that may be related to a user's intent. Query recommendation techniques are based either on clustering queries by their co-clicked URL patterns (Baeza-Yates et al., 2005) or on leveraging co-occurrences of sequential queries in web

search sessions (Zhang and Nasraoui, 2006; Boldi et al., 2008; Sadikov et al., 2010). The key difference from smart selection is that in our task the output is a selection that is relevant to the context of the document where the original selection appears (e.g., by adding terms neighboring the selection). In query recommendation, however, there is no notion of a document being read by the user and, instead, the recommendations are based exclusively on the aggregation of behavior of multiple users.

3 Problem Setting and Data

3.1 Smart Selection Definition

Let $\mathbf{D}$ be the set of all documents. We define a *selection* to be a character $\langle$offset, length$\rangle$-tuple in a document $d \in \mathbf{D}$. Let $\mathbf{S}$ be the set of all possible selections in $\mathbf{D}$ and let $\mathbf{S}_d$ be the set of all possible selections in d.

We define a scored smart selection, σ, in a document d, as a pair $\sigma = \langle x, y \rangle$ where $x \in \mathbf{S}_d$ is a selection and $y \in \mathbb{R}^+$ is a score for the selection.

We formally define the smart selection function ϕ as producing a ranked scored list of all possible selections from a document and user selection pair [2]:

$$\phi : \mathbf{D} \times \mathbf{S} \rightarrow (\sigma_1, ..., \sigma_{|\mathbf{S_d}|} \mid x_i \in \mathbf{S}_d, y_i \geq y_{i+1}) \quad (1)$$

Consider a user who selects s in a document d. Let τ be the target selection that best captures what the user intended to select. We define the smart selection task as recovering τ given the pair $\langle d, s \rangle$. Our problem then is to learn a function ϕ that best recovers the target selection from any user selection.

Note that even for a human, reconstructing an intended selection from a single word selection is not trivial. While there are some fairly clear cut cases such as expanding the selection "Obama" to `Barack Obama` in the sentence "*While in DC, Barack Obama met with...*", there are cases where the user intention depends on extrinsic factors such as the user's interests. For example, in a phrase "*University of California at Santa Cruz*" with a selection "California", some (albeit probably few) users may indeed be interested in the state of `California`, others in the `University of California` system of universities, and yet others specifically in the `University of California at Santa Cruz`. In the next section, we describe how we obtained a dataset of true intended user selections.

[2] The output consists of a ranked list of selections instead of a single selection to allow experiences such as proposing an n-best list to the user.

3.2 Data

In order to obtain a representative dataset for the smart selection task, we focus on a real-world application of users interacting with a touch-enabled e-reader device. In this application, a user is reading a book and chooses phrases for which she would like to get information from resources such as a dictionary, Wikipedia, or web search. Yet, because of the touch interface, she may only touch on a single word.

3.2.1 Crowdsourced Intended Selections

We obtain the intended selections through the following crowdsourcing exercise. We use the entire collection of textbooks in English from Wikibooks[3], a repository of publicly available textbooks. The corpus consists of 2,696 textbooks that span a large variety of categories such as Computing, Humanities, Science, etc. We first produce a uniform random sample of 100 books, and then sample one paragraph from each book. The resulting set of 100 paragraphs is then sent to the crowdsourcing system. Each paragraph is evaluated by 100 judges, using a pool of 152 judges. For each paragraph, we request the judges to select complete phrases for which they would like to "learn more in resources such as Wikipedia, search engines and dictionaries", i.e., our true user intended selections. As a result of this exercise, we obtain 13,681 judgments, corresponding to 4,067 unique intended selections. The distribution of number of unique judges who selected each unique intended selection, in a log-log scale, is shown in Figure 1. Notice that this is a Zipfian distribution since it follows a linear trend in the log-log scale.

Intuitively, the likelihood that a phrase is of interest to a user correlates with the number of judges who select that phrase. We thus use the number of judges who selected each phrase as a proxy for the likelihood that the phrase will be chosen by users.

The resulting dataset consists of 4,067 $\langle d, \tau \rangle$-pairs where d is a Wikibook document paragraph and τ is an intended selection, along with the number of judges who selected it. We further assigned

[3] Available at http://wikibooks.org.

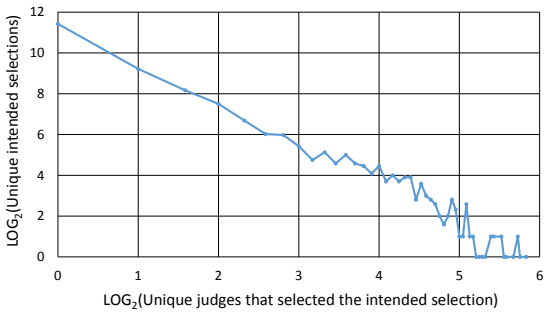


Figure 1: Zipfian distribution of unique intended selections vs. the number of judges who selected them, in log-log scale.

each pair to one of five randomly chosen folds, which are used for cross-validation experiments.

3.2.2 Testset Construction

We define a test case as a triple $\langle d, s, \tau \rangle$ where s is a simulated user selection. For each $\langle d, \tau \rangle$-pair in our dataset we construct n corresponding test cases by simulating the user selections $\{\langle d, \tau, s_1 \rangle, \ldots, \langle d, \tau, s_n \rangle\}$ where $s_1, \ldots, s_n$ correspond to the individual words in τ. In other words, each word in τ is considered as a candidate user selection.

We discard all target selections that only a single judge annotated since we observed that these mostly contained errors and noise, such as full sentences or nonsensical long sentence fragments.

Our first testset, labeled $\mathbf{T}_{\texttt{ALL}}$, is the resulting traffic-weighted multiset. That is, each test case $\langle d, s, \tau \rangle$ appears k times, where k is the number of judges who selected τ in d. $\mathbf{T}_{\texttt{ALL}}$ consists of 33,913 test cases.

We further utilize the distribution of judgments in the creation of three other testsets. Following the stratified sampling methodology commonly employed in the IR community, we construct testsets for the frequently, less frequently, and rarely annotated intended selections, which we call HEAD, TORSO, and TAIL, respectively. We obtain these testsets by first sorting each unique selection according to their frequency of occurrence, and then partitioning the set so that HEAD corresponds to the elements at the top of the list that account for 20% of the judgments; TAIL corresponds to the elements at the bottom also accounting for 20% of the judgments; and TORSO corresponds to the remaining elements. The resulting test sets, $\mathbf{T}_{\texttt{HEAD}}$, $\mathbf{T}_{\texttt{TORSO}}$, $\mathbf{T}_{\texttt{TAIL}}$ consist of 114, 2115, and 5798 test cases, respectively[4].

Test sets along with fold assignments and annotation guidelines are available at http://research.microsoft.com/en-us/downloads/eb42522c-068e-404c-b63f-cf632bd27344/.

3.3 Discussion

Our focus on single word selections is motivated by the touchscreen scenario presented in Section 1. Although our touch simulation assumes that each word in a target selection is equally likely to be selected by a user, in fact we expect this distribution to be non-uniform. For example, users may tend to select the first or last word more frequently than words in the middle of the target selection. Or perhaps users tend to select nouns and verbs more frequently than function words. We consider this out of scope for our paper, but view it as an important avenue of future investigation. Finally, for non-touchscreen environments, such as the desktop case, it would also be interesting to study the problem on multi-word user selections.

To get an idea of the kind of intended selections that comprise our dataset, we broke them down according to whether they referred to named entities or not. Perhaps surprisingly, the fraction of named entities in the dataset is quite low, 24.3%[5]. The rest of the intended selections mostly correspond to concepts and topics such as *embouchure formation*, *vocal fold relaxation*, *NHS voucher values*, *time-domain graphs*, etc.

4 Model

As argued in Section 1, existing techniques, such as NER taggers, chunkers, Knowledge Base lookup, etc., are geared towards *aspects* of the task (i.e., NEs, concepts, KB entries), but not the task as a whole. We can, however, combine the outputs of these systems with a learned "meta-model". The meta-model ranks the combined candidates according to a criterion that is derived from data that resembles real usage of smart selection as closely as possible. This technique is known

[4] We stress that $\mathbf{T}_{\texttt{ALL}}$ is a multi-set, reflecting the overall expected user traffic from our 100 judges per paragraph. $\mathbf{T}_{\texttt{HEAD}}$, $\mathbf{T}_{\texttt{TORSO}}$, $\mathbf{T}_{\texttt{TAIL}}$, in contrast, are not multi-sets since judgment frequency is already accounted for in the stratification process, as commonly done in the IR community.

[5] Becker et al. (2012) report a similar finding, showing that only 26% of questions, which a user might ask after reading a Wikipedia article, are focused on named entities.

in the machine learning community as *ensemble learning* (Dietterich, 1997).

Our ensemble approach, described in this section, serves as our main implementation of the smart selection function ϕ of Equation 1. Each of the ensemble members are themselves a separate implementation of ϕ and will be used as a point of comparison in our experiments. Below, we describe the ensemble members before turning to the ensemble learner.

4.1 Ensemble Members

4.1.1 Hyperlink Intent Model

The Hyperlink Intent Model (HIM), which leverages web graph information, is a machine-learned system based on the intuition that anchor texts in Wikipedia are good representations of what users might want to learn about. We build upon the fact that Wikipedia editors write anchor texts for entities, concepts, and things of potential interest for follow-up to other content. HIM learns to recover anchor texts from their single word subselections.

Specifically, HIM iteratively decides whether to expand the current selection (initially a single word) one word to the left or right via greedy binary decisions, until a stopping condition is met. At each step, two binary classifiers are consulted. The first one scores the left expansion decision and the second one scores the right expansion decision. In addition, we use the same two classifiers to evaluate the expansion decision "from the outside in", i.e., from the word next to the current selection (left and right, respectively) to the closest word in the current selection. If the probability for expansion of any model exceeds a predefined threshold, then the most probable expansion is chosen and we continue the iteration with the newly expanded selection as input. The algorithm is illustrated in Figure 2.

We automatically create our training set for HIM by first taking a random sample of 8K Wikipedia anchor texts. We treat each anchor text as an intended selection, and each word in the anchor text as a simulated user selection. For each word to the left (or the right) of the user selection that is part of the anchor text, we create a positive training example. Similarly, for each word to the left (or the right) that is outside of the anchor text, we create a negative training example. We include additional negative examples using random word selections from Wikipedia content. For this purpose we sample random words that are not part of an anchor text. Our final data consists of 2.6M data points, with a 1:20 ratio of positive to negative examples[6].

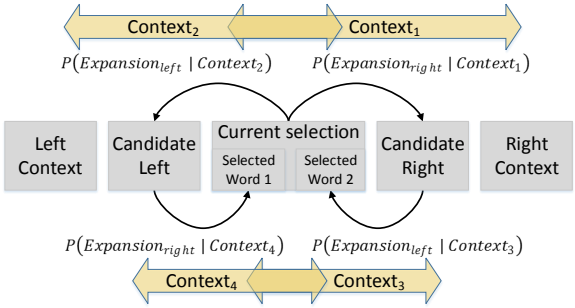


Figure 2: Hyperlink Intent Model (HIM) decoding flow for smart selection.

We use logistic regression as the classification algorithm for our binary classifiers. The features used by each model are computed over three strings: the current selection s (initially the single-word simulated user selection), the candidate expansion word w, and one word over from the right or left of s. The features fall into five feature families: (1) *character-level features*, including capitalization, all-cap formatting, character length, presence of opening/closing parentheses, presence and position of digits and non-alphabetic characters, and minimum and average character uni/bi/trigram frequencies (based on frequency tables computed offline from Wikipedia article content); (2) *stopword features*, which indicate the presence of a stop word (from a stop word list); (3) *tf.idf scores* precomputed from Wikipedia content statistics; (4) *knowledge base features*, which indicate whether a string matches an item or a substring of an item in the knowledge base described in Section 4.1.2 below; and (5) *lexical features*, which capture the actual string of the current selection and the candidate expansion word.

4.1.2 Unit Spotting

Our second qualitative class of ensemble members use notions of *unit* that are either based on linguistic constituency or knowledge base presence. The general process is that any unit that subsumes the user selection is treated as a smart selection candidate. Scoring of candidates is by normalized length, under the assumption that in general the most specific (longest) unit is more likely to be the intended selection.

[6]Note that this training set is generated automatically and is, by design, of a different nature than the manually labeled data we use to train and test the ensemble model.

Our first unit spotter, labeled NER is geared towards recognizing named entities. We use a commercial and proprietary state-of-the-art NER system, trained using the perceptron algorithm (Collins, 2002) over more than a million hand-annotated labels.

Our second approach uses purely syntactic information and treats noun phrases as units. We label this model as NP. For this purpose we parse the sentence containing the user selection with a syntactic parser following (Ratnaparkhi, 1999). We then treat every noun phrase that subsumes the user selection as a candidate smart selection.

Finally, our third unit spotter, labeled KB, is based on the assumption that concepts and other entries in a knowledge base are, by nature, things that can be of interest to people. For our knowledge base lookup, we use a proprietary graph consisting of knowledge from Wikipedia, Freebase, and paid feeds from various providers from domains such as entertainment, local, and finance.

4.1.3 Heuristics

Our third family of ensemble members implements simple heuristics, which tend to be high precision especially in the HEAD of our data.

The first heuristic, representing the current touch-enabled selection paradigm seen in many of today's tablets and smartphones, is labeled CUR. It simply assumes that the intended selection is always the user-selected word.

The second is a capitalization-based heuristic (CAP), which simply expands every selected capitalized word selection to the longest uninterrupted sequence of capitalized words.

4.2 Ensemble Learning

In this section, we describe how we train our meta-learner, labeled ENS, which takes as input the candidate lists produced by the ensemble members from Section 4.1, and scores each candidate, producing a final scored ranked list.

We use logistic regression as a classification algorithm to address this task. Our 22 features in ENS consist of three main classes: (1) features related to the individual ensemble members; (2) features related to the user selection; and (3) features related to the candidate smart selection. For (1), the features consist of whether a particular ensemble member generated the candidate smart selection and its score for that candidate. If the candidate smart selection is not in the candidate list of an ensemble member, its score is set to zero. For both (2) and (3), features account for length and capitalization properties of the user selection and the candidate smart selection (e.g., token length, ratio of capitalized tokens, ratio of capitalized characters, whether or not the first and last tokens are capitalized.)

Although training data for the HIM model was automatically generated from Wikipedia, for ENS we desire training data that reflects the true expected user experience. For this, we use five-fold cross-validation over our data collection described in Section 3.2. That is, to decode a fold with our meta-learner, we train ENS with the other four folds. Note that every candidate selection for a $\langle$document, user selection$\rangle$-pair, $\langle d, s\rangle$, for the same d and s, are assigned to a single fold, hence the training process does not see any user selection from the test set.

5 Experimental Results

5.1 Experimental Setup

Recall our testsets $\mathbf{T}_{\texttt{ALL}}$, $\mathbf{T}_{\texttt{HEAD}}$, $\mathbf{T}_{\texttt{TORSO}}$, and $\mathbf{T}_{\texttt{TAIL}}$ from Section 3.2.2, where a test case is defined as a triple $\langle d, s, \tau\rangle$, and where d is a document, s is a user selection, and τ is the intended user selection. In this section, we describe our evaluation metric and summarize the system configurations that we evaluate.

5.1.1 Metric

In our evaluation, we apply the smart selection function $\phi(d, s)$ (see Eq. 1) to each test case and measure how well it recovers τ.

Let $\mathbf{A}$ be the set of $\langle d, \tau\rangle$-pairs from our dataset described in Section 3.2.1 that corresponds to a testset $\mathbf{T}$. Let $\mathbf{T}_{\langle d,\tau\rangle}$ be the set of all test cases in $\mathbf{T}$ with a fixed d and τ. We define the macro precision of a smart selection function, P_ϕ, as follows:

$$P_\phi = \frac{1}{\mid \mathbf{A} \mid} \sum_{\langle d,\tau\rangle \in \mathbf{A}} P_\phi(d, \tau) \qquad (2)$$

$$P_\phi(d, \tau) = \frac{1}{\mid \mathbf{T}_{\langle d,\tau\rangle} \mid} \sum_{\langle d,s,\tau\rangle \in \mathbf{T}_{\langle d,\tau\rangle}} P_\phi(d, s, \tau)$$

$$P_\phi(d, s, \tau) = \frac{1}{\mid \phi(d, s) \mid} \sum_{\sigma \in \phi(d,s)} I(\sigma, \tau)$$

$$I(\sigma, \tau) = \begin{cases} 1 & \text{if } \sigma = \langle x, y\rangle \wedge x = \tau \\ 0 & \text{otherwise} \end{cases}$$

	CP@1	CP@2	CP@3	CP@4	CP@5
CUR	39.3	-	-	-	-
CAP	48.9	51.0	51.2	51.8	51.8
NER	43.5	-	-	-	-
NP	34.1	50.2	55.5	57.1	57.6
KB	50.2	50.8	50.9	50.9	50.9
HIM	48.1	48.8	48.8	48.8	48.8
ENS	$56.8^{\dagger}$	$76.0^{\ddagger}$	$82.6^{\ddagger}$	$85.2^{\ddagger}$	$86.6^{\ddagger}$

Table 1: Smart selection performance, as a function of CP, on $\mathbf{T}_{\texttt{ALL}}$. ‡ and † indicate statistical significance with $p = 0.01$ and 0.05, respectively. An oracle ensemble would achieve an upper bound CP of 87.3%.

We report cumulative macro precision at rank (**CP@k**) in our experiments since our testsets contain a single true user-intended selection for each test case[7]. However, this is an overly conservative metric since in many cases an alternative smart selection might equally please the user. For example, if our testset contains a user intended selection τ = `The University of Southern California`, then given the simulated selection "California", both τ and `University of Southern California` would most likely equally satisfy the user intent (whereas the latter would be considered incorrect in our evaluation). In fact, the ideal testset would further evaluate the *distance* or *relevance* of the smart selection to the intended user selection. We would then find perhaps that `Southern California` is a more reasonable smart selection than `of Southern California`. However, precisely defining such a *relevance* function and designing the guidelines for a user study is non-trivial and left for future work.

5.1.2 Systems

In our experiments, we evaluate the following systems, each described in detail in Section 4: Passthrough (CUR), Capitalization (CAP), Named-Entity Recognizer (NER), Noun Phrase (NP), Knowledge Base (KB), Hyperlink Intent Model (HIM), Ensemble (ENS).

5.2 Results

Table 1 reports the smart selection performance on the full traffic weighted testset $\mathbf{T}_{\texttt{ALL}}$, as a function of CP@$k$. Our ensemble approach recovers the true user-intended selection in 56.8% of the cases. In its top-2 and top-3 ranked smart selections, the true user-intended selection is retrieved 76.0% and 82.6% of the time, respectively. In position 1, ENS significantly outperforms all other systems with 95% confidence. Moreover, we notice that the divergence between ENS and the other systems greatly increases for $K \geq 2$, where the significance is now at the 99% level.

The CUR system models the selection paradigm of today's consumer touch-enabled devices (i.e., it assumes that the intented selection is always the touched word). Without changing the user interface, we report a 45% improvement in predicting what the user intended to select over this baseline. If we changed the user interface to allow two or three options to be displayed to the user, then we would improve by 93% and 110%, respectively.

For CUR and NER, we report results only at $K = 1$ since these systems only ever return a single smart selection. Note also that when no named entity is found by NER, or no noun phrase is found by NP or no knowledge base entry is found by KB, the corresponding systems return the original user selection as their smart selection.

CAP does not vary much across K: when the intended selection is a capitalized multi-word, the longest string tends to be the intended selection. The same holds for KB.

Whereas Table 1 reports the aggregate expected traffic performance, we further explore the performance against the stratified $\mathbf{T}_{\texttt{HEAD}}$, $\mathbf{T}_{\texttt{TORSO}}$, and $\mathbf{T}_{\texttt{TAIL}}$ testsets. The results are summarized in Table 2. As outlined in Section 3.2, the HEAD selections tend to be disproportionately entities and capitalized terms when compared to the TORSO and TAIL. Hence CAP, NER and KB perform much better on the HEAD. In fact, on the HEAD, CAP performs statistically as well as the ENS model. This means that at position 1, for systems that need to focus only on the HEAD, a very simple solution is adequate. For TORSO and TAIL, however, ENS performs better. At positions 2 and 3, across all strata, the ENS model significantly outperforms all other systems (with 99% confidence).

Next, we studied the relative contribution of each ensemble member to the ENS model. Figure 3 illustrates the results of the ablation study. The ensemble member that results in the biggest performance drop when removed is HIM. Perhaps

[7]Because there is only a single true intended selection for each test case, Recall@k = CP@k.

	HEAD			TORSO			TAIL		
	CP@1	**CP@2**	**CP@3**	**CP@1**	**CP@2**	**CP@3**	**CP@1**	**CP@2**	**CP@3**
CUR	48.5	-	-	36.7	-	-	26.6	-	-
CAP	74.2	74.7	74.8	43.0	45.0	45.1	26.1	27.4	28.2
NER	60.6	-	-	39.2	-	-	26.7	-	-
NP	52.3	64.9	69.4	31.0	48.2	53.8	20.0	32.2	35.7
KB	66.7	66.7	66.7	47.0	47.9	48.1	29.9	30.1	30.1
HIM	64.4	65.7	65.7	44.7	45.2	45.4	27.9	28.2	28.2
ENS	75.8	$91.8^{\ddagger}$	$96.5^{\ddagger}$	$52.7^{\dagger}$	$73.7^{\ddagger}$	$81.5^{\ddagger}$	$32.4^{\dagger}$	$50.7^{\ddagger}$	$58.5^{\ddagger}$

Table 2: Smart selection performance, as a function of CP, on the $\mathbf{T}_{\texttt{HEAD}}$, $\mathbf{T}_{\texttt{TORSO}}$, and $\mathbf{T}_{\texttt{TAIL}}$ testsets. ‡ and † indicate statistical significance with $p = 0.01$ and 0.05, respectively. An oracle ensemble would achieve an upper bound CP of 98.5%, 86.8% and 64.8% for $\mathbf{T}_{\texttt{HEAD}}$, $\mathbf{T}_{\texttt{TORSO}}$, and $\mathbf{T}_{\texttt{TAIL}}$, respectively.

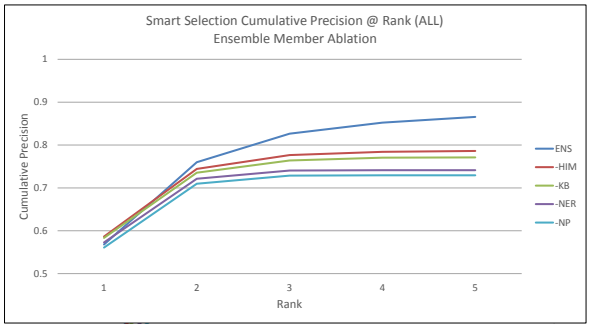


Figure 3: Ablation of ensemble model members over $\mathbf{T}_{\texttt{ALL}}$. Each consecutive model removes one member specified in the series name.

surprisingly, a first ablation of either the CAP or KB model, two of the better individual performing models from Table 1, leads to an ablated-ENS performance that is nearly identical to the full ENS model. One possible reason is that both tend to generate similar candidates (i.e., many entities in our KB are capitalized). Although the HIM model as a standalone system does not outperform simple linguistic unit selection models, it appears to be the most important contributor to the overall ensemble.

5.3 Error Analysis: Oracle Ensemble

We begin by assessing an upper bound for our ensemble, i.e., an *oracle ensemble*, by assuming that if a correct candidate is generated by any ensemble member, the oracle ensemble model places it in first position. For $\mathbf{T}_{\texttt{ALL}}$ the oracle performance is 87.3%. In other words, our choice of ensemble members was able to recover a correct smart selection as a candidate in 87.3% of the user study cases. For $\mathbf{T}_{\texttt{HEAD}}$, $\mathbf{T}_{\texttt{TORSO}}$, and $\mathbf{T}_{\texttt{TAIL}}$, the oracle performance is 98.5%, 86.8%, and 64.8%, respectively.

Although our ENS model's CP@3 is within 2-6 points of the oracle, there is room to significantly improve our CP@1, see Table 1 and Table 2. We analyze this opportunity by inspecting a random sample of 200 test cases where ENS produced an incorrect smart selection in position 1. The breakdown of these cases is: 1 case from $\mathbf{T}_{\texttt{HEAD}}$; 50 cases from $\mathbf{T}_{\texttt{TORSO}}$; 149 cases from $\mathbf{T}_{\texttt{TAIL}}$, i.e., most errors occur in the TAIL.

For 146 of these cases (73%), not a single ensemble member produced the correct target selection τ as a candidate. We analyze these cases in detail in Section 5.4. Of the remaining cases, 25, 10, 9, 4, 4, and 2 were correct in positions 2, 3, 4, 5, 6, 7, respectively. Table 3 lists some examples.

In 18 cases (33%), the result in position 1 is very reasonable given the context and user selection (see lines 1-4 in Table 3 for examples). Often the target selection was also found in second position. These cases highlight the need for a more relaxed, relevance-based user study, as pointed out at the end of Section 5.1.1.

We attributed 7 (13%) of the cases to data problems: some cases had a punctuation as a sole character user selection, some had a mishandled escaped quotation character, and some had a UTF-8 encoding error.

The remaining 29 (54%) were truly model errors. Some examples are shown in lines 5-8 in Table 3. We found three categories of errors here. First, our model has learned a strong prior on preferring the original user selection (see example line 5). From a user experience point of view, when the model is unsure of itself, it is in fact better not to alter her selection. Second, we also learned a strong capitalization prior, i.e., to trust the CAP member (see example line 6). Finally, we noticed that we have difficulty handling user selections consisting of a stopword (we noted determiners, prepositions, and the word "and"). Adding a few simple features to ENS based on a stopwords list or a list of closed-class words should address this problem.

	Text Snippet	User Selection	ENS 1st Result
1	"The **Russian conquest of the South Caucasus in the 19th century** split the speech community across two states..."	Caucasus	South Caucasus
2	"...are generally something that **transportation** agencies would like to minimize..."	transportation	transportation agencies
3	"The **vocal organ of birds**, the syrinx, is located at the base of the blackbird's trachea."	vocal	vocal organ
4	"An example of this may be an idealised **waveform** like a square wave..."	waveform	idealised waveform
5	"Tickets may be purchased from either the **ticket counter** or from automatic machines..."	counter	counter
6	"PBXT features include the following: **MVCC** Support: MVCC stands for Multi-version Concurrency Control."	MVCC	MVCC Support
7	"Centers for song production pathways include the High vocal center; **robust nucleus of archistriatum** (RA); and the tracheosyringeal part of the hypoglossal nucleus..."	robust	robust nucleus
8	"...and get an **11gR2 RAC cluster database** running inside virtual machines..."	cluster	RAC cluster

Table 3: Position 1 errors when applying ENS to our test cases. The text snippet is a substring of a paragraph presented to our judges with the target selection (τ) indicated in bold.

5.4 Error Analysis: Ensemble Members

Over all test cases, the distribution of cases without a correct candidate generated by an ensemble member in the HEAD, TORSO, TAIL is 0.3%, 34.6%, and 65.1%, respectively. We manually inspected a random sample of 100 such test cases.

The majority of them, 83%, were large sentence fragments, which we consider out of scope according to our prediction task definition outlined in Section 1. The average token length of the target selection τ for these was 15.3. In comparison, we estimate the average token length of the task-admissable cases to be 2.7 tokens. Although most of these long fragment selections seem to be noise, a few cases are statements that a user would reasonably want to know more about, such as: (i) "Talks of a merger between the NHL and the WHA were growing" or (ii) "NaN + NaN * 1.0i".

In 10% of the cases, we face a punctuation-handling issue, and in each case our ensemble was able to generate a correct candidate when fixing the punctuation. For example, for the book title $\tau =$ What is life?, our ensemble found the candidate What is life, dropping the question mark. For $\tau =$ Near Earth Asteroid. our ensemble found Near Earth Asteroid, dropping the period. Similar problems occurred with parentheses and quotation marks.

In two cases, our ensemble members dropped a leading "the" token, e.g., for $\tau =$ the Hume Highway, we found Hume Highway.

Finally, 2 cases were UTF-8 encoding mistakes, leaving five "true error" cases.

6 Conclusion and Future Work

We introduced a new paradigm, *smart selection*, to address the cumbersome text selection capabilities of today's touch-enabled mobile devices. We report 45% improvement in predicting what the user intended to select over current touch-enabled consumer platforms, such as iOS, Android and Windows. We release to the community a dataset of $33,912$ crowdsourced true intended user selections and corresponding simulated user touches.

There are many avenues for future work, including understanding the distribution of user touches on their intended selection, other interesting scenarios (e.g., going beyond the e-reader towards document editors and web browsers may show different distributions in what users select), leveraging other sources of signal such as a user's profile, her interests and her local session context, and exploring user interfaces that leverage n-best smart selection prediction lists, for example by providing selection options to the user after her touch.

With the release of our $33,912$-crowdsourced dataset and our model analyses, it is our hope that the research community can help accelerate the progress towards reinventing the way text selection occurs today, the initial steps for which we have taken in this paper.

7 Acknowledgments

The authors thank Aitao Chen for sharing his NER tagger for our experiments, and Bernhard Kohlmeier, Pradeep Chilakamarri, Ashok Chandra, David Hamilton, and Bo Zhao for their guidance and valuable discussions.

References

Steven. P. Abney. 1991. Parsing by chunks. In Robert C. Berwick, Steven P. Abney, and Carol Tenny, editors, *Principle-Based Parsing: Computation and Psycholinguistics*, pages 257–278. Kluwer, Dordrecht.

Ricardo Baeza-Yates, Carlos Hurtado, and Marcelo Mendoza. 2005. Query recommendation using query logs in search engines. In *Current Trends in Database Technology-EDBT 2004 Workshops*, pages 588–596. Springer.

Lee Becker, Sumit Basu, and Lucy Vanderwende. 2012. Mind the gap: Learning to choose gaps for question generation. In *Proceedings of NAACL HLT '12*, pages 742–751.

Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, Aristides Gionis, and Sebastiano Vigna. 2008. The query-flow graph: model and applications. In *Proceedings of CIKM '08*, pages 609–618. ACM.

Nancy A. Chinchor. 1998. Named entity task definition. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, Fairfax, VA.

Michael Collins. 2002. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*.

Thomas G. Dietterich. 1997. Machine Learning Research - Four Current Directions. *AI Magazine*, 18:4:97–136.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *In ACL*, pages 363–370.

Asela Gunawardana, Tim Paek, and Christopher Meek. 2010. Usability guided key-target resizing for soft keyboards. In *Proceedings of IUI '10*, pages 111–118.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *Proceedings of NAACL '01*, pages 1–8.

Anuj Kumar, Tim Paek, and Bongshin Lee. 2012. Voice typing: A new speech interaction model for dictation on touchscreen devices. In *Proceedings of CHI'12*, pages 2277–2286.

Marcia Muñoz, Vasin Punyakanok, Dan Roth, and Dav Zimak. 1999. A learning approach to shallow parsing. In *Proceedings of EMNLP/VLC*, pages 168–178.

Lance A. Ramshaw and Mitchell P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the 3rd ACL Workshop on Very Large Corpora*, pages 82–94. Cambridge MA, USA.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of CoNLL-2009*, pages 147–155.

Adwait Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Mach. Learn.*, 34(1-3):151–175, February.

Eldar Sadikov, Jayant Madhavan, Lu Wang, and Alon Halevy. 2010. Clustering query refinements by user intent. In *Proceedings of the 19th international conference on World wide web*, pages 841–850. ACM.

Daisuke Sakamoto, Takanori Komatsu, and Takeo Igarashi. 2013. Voice augmented manipulation: using paralinguistic information to manipulate mobile devices. In *Mobile HCI*, pages 69–78.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada.

Zhiyong Zhang and Olfa Nasraoui. 2006. Mining search engine query logs for query recommendation. In *Proceedings of the 15th international conference on World Wide Web*, pages 1039–1040. ACM.

Modeling Prompt Adherence in Student Essays

Isaac Persing and **Vincent Ng**
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
{persingq,vince}@hlt.utdallas.edu

Abstract

Recently, researchers have begun exploring methods of scoring student essays with respect to particular dimensions of quality such as coherence, technical errors, and prompt adherence. The work on modeling prompt adherence, however, has been focused mainly on whether individual sentences adhere to the prompt. We present a new annotated corpus of essay-level prompt adherence scores and propose a feature-rich approach to scoring essays along the prompt adherence dimension. Our approach significantly outperforms a knowledge-lean baseline prompt adherence scoring system yielding improvements of up to 16.6%.

1 Introduction

Automated essay scoring, the task of employing computer technology to evaluate and score written text, is one of the most important educational applications of natural language processing (NLP) (see Shermis and Burstein (2003) and Shermis et al. (2010) for an overview of the state of the art in this task). A major weakness of many existing scoring engines such as the Intelligent Essay Assessor^{TM}(Landauer et al., 2003) is that they adopt a holistic scoring scheme, which summarizes the quality of an essay with a single score and thus provides very limited feedback to the writer. In particular, it is not clear which dimension of an essay (e.g., style, coherence, relevance) a score should be attributed to. Recent work addresses this problem by scoring a particular dimension of essay quality such as coherence (Miltsakaki and Kukich, 2004), technical errors, organization (Persing et al., 2010), and thesis clarity (Persing and Ng, 2013). Essay grading software that provides feedback along multiple dimensions of essay quality such as E-*rater*/Criterion (Attali and Burstein, 2006) has also begun to emerge.

Our goal in this paper is to develop a computational model for scoring an essay along an under-investigated dimension — *prompt adherence*. Prompt adherence refers to how related an essay's content is to the prompt for which it was written. An essay with a high prompt adherence score consistently remains on the topic introduced by the prompt and is free of irrelevant digressions.

To our knowledge, little work has been done on scoring the prompt adherence of student essays since Higgins et al. (2004). Nevertheless, there are major differences between Higgins et al.'s work and our work with respect to both the way the task is formulated and the approach. Regarding task formulation, while Higgins et al. focus on classifying each *sentence* as having either *good* or *bad* adherence to the prompt, we focus on assigning a prompt adherence score to the entire *essay*, allowing the score to range from one to four points at half-point increments. As far as the approach is concerned, Higgins et al. adopt a *knowledge-lean* approach to the task, where almost all of the features they employ are computed based on a word-based semantic similarity measure known as *Random Indexing* (Kanerva et al., 2000). On the other hand, we employ a large variety of features, including lexical and knowledge-based features that encode how well the concepts in an essay match those in the prompt, LDA-based features that provide semantic generalizations of lexical features, and "error type" features that encode different types of errors the writer made that are related to prompt adherence.

In sum, our contributions in this paper are twofold. First, we develop a scoring model for the prompt adherence dimension on student essays using a feature-rich approach. Second, in order to stimulate further research on this task, we make our data set consisting of prompt adherence an-

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1534–1543,
Baltimore, Maryland, USA, June 23-25 2014.

Topic	Languages	Essays
Most university degrees are theoretical and do not prepare students for the real world. They are therefore of very little value.	13	131
The prison system is outdated. No civilized society should punish its criminals: it should rehabilitate them.	11	80
In his novel *Animal Farm*, George Orwell wrote "All men are equal but some are more equal than others." How true is this today?	10	64

Table 1: Some examples of writing topics.

notations of 830 essays publicly available. Since progress in prompt adherence modeling is hindered in part by the lack of a publicly annotated corpus, we believe that our data set will be a valuable resource to the NLP community.

2 Corpus Information

We use as our corpus the 4.5 million word International Corpus of Learner English (ICLE) (Granger et al., 2009), which consists of more than 6000 essays written by university undergraduates from 16 countries and 16 native languages who are learners of English as a Foreign Language. 91% of the ICLE texts are argumentative. We select a subset consisting of 830 argumentative essays from the ICLE to annotate for training and testing of our essay prompt adherence scoring system. Table 1 shows three of the 13 topics selected for annotation. Fifteen native languages are represented in the set of annotated essays.

3 Corpus Annotation

We ask human annotators to score each of the 830 argumentative essays along the prompt adherence dimension. Our annotators were selected from over 30 applicants who were familiarized with the scoring rubric and given sample essays to score. The six who were most consistent with the expected scores were given additional essays to annotate. Annotators evaluated how well each essay adheres to its prompt using a numerical score from one to four at half-point increments (see Table 2 for a description of each score). This contrasts with previous work on prompt adherence essay scoring, where the corpus is annotated with a binary decision (i.e., *good* or *bad*) (e.g., Higgins et al. (2004; 2006), Louis and Higgins (2010)). Hence, our annotation scheme not only provides a finer-grained distinction of prompt adherence (which can be important in practice), but also makes the prediction task more challenging.

Score	Description of Prompt Adherence
4	essay fully addresses the prompt and **consistently stays on topic**
3	essay mostly addresses the prompt or **occasionally wanders off topic**
2	essay does not fully address the prompt or **consistently wanders off topic**
1	essay does not address the prompt at all or is **completely off topic**

Table 2: Descriptions of the meaning of scores.

To ensure consistency in annotation, we randomly select 707 essays to have graded by multiple annotators. Analysis reveals that the Pearson's correlation coefficient computed over these doubly annotated essays is 0.243. Though annotators exactly agree on the prompt adherence score of an essay only 38% of the time, the scores they apply fall within 0.5 points in 66% of essays and within 1.0 point in 89% of essays. For the sake of our experiments, whenever annotators disagree on an essay's prompt adherence score, we assign the essay the average of all annotations rounded to the nearest half point. Table 3 shows the number of essays that receive each of the seven scores for prompt adherence.

score	1.0	1.5	2.0	2.5	3.0	3.5	4.0
essays	0	0	8	44	105	230	443

Table 3: Distribution of prompt adherence scores.

4 Score Prediction

In this section, we describe in detail our system for predicting essays' prompt adherence scores.

4.1 Model Training and Application

We cast the problem of predicting an essay's prompt adherence score as 13 regression problems, one for each prompt. Each essay is represented as an instance whose label is the essay's true score (one of the values shown in Table 3) with up to seven types of features including baseline (Section 4.2) and six other feature types proposed by us (Section 4.3). Our regressors may assign an essay any score in the range of $1.0-4.0$.

Using regression captures the fact that some pairs of scores are more similar than others (e.g., an essay with a prompt adherence score of 3.5 is more similar to an essay with a score of 4.0 than it is to one with a score of 1.0). A classification sys-

tem, by contrast, may sometimes believe that the scores 1.0 and 4.0 are most likely for a particular essay, even though these scores are at opposite ends of the score range.

Using a different regressor for each prompt captures the fact that it may be easier for an essay to adhere to some prompts than to others, and common problems students have writing essays for one prompt may not apply to essays written in response to another prompt. For example, in essays written in response to the prompt "Marx once said that religion was the opium of the masses. If he was alive at the end of the 20th century, he would replace religion with television," students sometimes write essays about all the evils of television, forgetting that their essay is only supposed to be about whether it is "the opium of the masses". Students are less likely to make an analogous mistake when writing for the prompt "Crime does not pay."

After creating training instances for prompt p_i, we train a linear regressor, r_i, with regularization parameter c_i for scoring test essays written in response to p_i using the linear SVM regressor implemented in the LIBSVM software package (Chang and Lin, 2001). All SVM-specific learning parameters are set to their default values except c_i, which we tune to maximize performance on held-out validation data.

After training the classifiers, we use them to classify the test set essays. The test instances are created in the same way as the training instances.

4.2 Baseline Features

Our baseline system for score prediction employs various features based on Random Indexing.

1. Random Indexing Random Indexing (RI) is "an efficient, scalable and incremental alternative" (Sahlgren, 2005) to Latent Semantic Indexing (Deerwester et al., 1990; Landauer and Dutnais, 1997) which allows us to automatically generate a semantic similarity measure between any two words. We train our RI model on over 30 million words of the English Gigaword corpus (Parker et al., 2009) using the S-Space package (Jurgens and Stevens, 2010). We expect that features based on RI will be useful for prompt adherence scoring because they may help us find text related to the prompt even if some of its concepts have have been rephrased (e.g., an essay may talk about "jail" rather than "prison", which is mentioned in one of the prompts), and because they have already proven useful for the related task of determining which sentences in an essay are related to the prompt (Higgins et al., 2004).

For each essay, we therefore attempt to adapt the RI features used by Higgins et al. (2004) to our problem of prompt adherence scoring. We do this by generating one feature encoding the entire essay's similarity to the prompt, another encoding the essay's highest individual sentence's similarity to the prompt, a third encoding the highest entire essay similarity to one of the prompt sentences, another encoding the highest individual sentence similarity to an individual prompt sentence, and finally one encoding the entire essay's similarity to a manually rewritten version of the prompt that excludes extraneous material (such as "In his novel Animal Farm, George Orwell wrote," which is introductory material from the third prompt in Table 1). Our RI feature set necessarily excludes those features from Higgins et al. that are not easily translatable to our problem since we are concerned with an entire essay's adherence to its prompt rather than with each of its sentences' relatedness to the prompt. Since RI does not provide a straightforward way to measure similarity between groups of words such as sentences or essays, we use Higgins and Burstein's (2007) method to generate these features.

4.3 Novel Features

Next, we introduce six types of novel features.

2. N-grams As our first novel feature, we use the 10,000 most important lemmatized unigram, bigram, and trigram features that occur in the essay. N-grams can be useful for prompt adherence scoring because they can capture useful words and phrases related to a prompt. For example, words and phrases like "university degree", "student", and "real world" are relevant to the first prompt in Table 1, so it is more likely that an essay adheres to the prompt if they appear in the essay.

We determine the "most important" n-gram features using information gain computed over the training data (Yang and Pedersen, 1997). Since the essays vary greatly in length, we normalize each essay's set of n-gram features to unit length.

3. Thesis Clarity Keywords Our next set of features consists of the keyword features we introduced in our previous work on essay thesis clarity scoring (Persing and Ng, 2013). Below we give an overview of these keyword features and motivate

why they are potentially useful for prompt adherence scoring.

The keyword features were formed by first examining the 13 essay prompts, splitting each into its component pieces. As an example of what is meant by a "component piece", consider the first prompt in Table 1. The components of this prompt would be "Most university degrees are theoretical", "Most university degrees do not prepare students for the real world", and "Most university degrees are of very little value."

Then the most important (primary) and second most important (secondary) words were selected from each prompt component, where a word was considered "important" if it would be a good word for a student to use when stating her thesis about the prompt. So since the lemmatized version of the third component of the second prompt in Table 1 is "it should rehabilitate they", "rehabilitate" was selected as a primary keyword and "society" as a secondary keyword.

Features are then computed based on these keywords. For instance, one thesis clarity keyword feature is computed as follows. The RI similarity measure is first taken between the essay and each group of the prompt's primary keywords. The feature then gets assigned the lowest of these values. If this feature has a low value, that suggests that the student ignored the prompt component from which the value came when writing the essay.

To compute another of the thesis clarity keyword features, the numbers of combined primary and secondary keywords the essay contains from each component of its prompt are counted. These numbers are then divided by the total count of primary and secondary features in their respective components. The greatest of the fractions generated in this way is encoded as a feature because if it has a low value, that indicates the essay's thesis may not be very relevant to the prompt.[1]

4. Prompt Adherence Keywords The thesis clarity keyword features described above were intended for the task of determining how clear an essay's thesis is, but since our goal is instead to determine how well an essay adheres to its prompt, it makes sense to adapt keyword features to our task rather than to adopt keyword features exactly as they have been used before. For this reason, we construct a new list of keywords for each prompt component, though since prompt adherence is more concerned with what the student says about the topics than it is with whether or not what she says about them is stated clearly, our keyword lists look a little different than the ones discussed above. For an example, we earlier alluded to the problem of students merely discussing all the evils of television for the prompt "Marx once said that religion was the opium of the masses. If he was alive at the end of the 20th century, he would replace religion with television." Since the question suggests that students discuss whether television is analogous to religion in this way, our set of prompt adherence keywords for this prompt contains the word "religion" while the previously discussed keyword sets do not. This is because a thesis like "Television is bad" can be stated very clearly without making any reference to religion at all, and so an essay with a thesis like this can potentially have a very high thesis clarity score. It should not, however, have a very high prompt adherence score, as the prompt asked the student to discuss whether television is like religion in a particular way, so religion should be at least briefly addressed for an essay to be awarded a high prompt adherence score.

Additionally, our prompt adherence keyword sets do not adopt the notions of primary and secondary groups of keywords for each prompt component, instead collecting all the keywords for a component into one set because "secondary" keywords tend to be things that are important when we are concerned with what a student is saying about the topic rather than just how clearly she said it.

We form two types of features from prompt adherence keywords. While both types of features measure how much each prompt component was discussed in an essay, they differ in how they encode the information. To obtain feature values of the first type, we take the RI similarities between the whole essay and each set of prompt adherence keywords from the prompt's components. This results in one to three features, as some prompts have one component while others have up to three.

We obtain feature values of the second type as follows. For each component, we count the number of prompt adherence keywords the essay contains. We divide this number by the number of prompt adherence keywords we identified from

[1] Space limitations preclude a complete listing of the thesis clarity keyword features. See our website at `http://www.hlt.utdallas.edu/~persingq/ICLE/` for the complete list.

the component. This results in one to three features since a prompt has one to three components.

5. LDA Topics A problem with the features we have introduced up to this point is that they have trouble identifying topics that are not mentioned in the prompt, but are nevertheless related to the prompt. These topics should not diminish the essay's prompt adherence score because they are at least related to prompt concepts. For example, consider the prompt "All armies should consist entirely of professional soldiers: there is no value in a system of military service." An essay containing words like "peace", "patriotism", or "training" are probably not digressions from the prompt, and therefore should not be penalized for discussing these topics. But the various measures of keyword similarities described above will at best not notice that anything related to the prompt is being discussed, and at worst, this might have effects like lowering some of the RI similarity scores, thereby probably lowering the prompt adherence score the regressor assigns to the essay. While n-gram features do not have exactly the same problem, they would still only notice that these example words are related to the prompt if multiple essays use the same words to discuss these concepts. For this reason, we introduce Latent Dirichlet Allocation (LDA) (Blei et al., 2003) features.

In order to construct our LDA features, we first collect all essays written in response to each prompt into its own set. Note that this feature type exploits unlabeled data: it includes all essays in the ICLE responding to our prompts, not just those in our smaller annotated 830 essay dataset. We then use the MALLET (McCallum, 2002) implementation of LDA to build a topic model of 1,000 topics around each of these sets of essays. This results in what we can think of as a soft clustering of words into 1,000 sets for each prompt, where each set of words represents one of the topics LDA identified being discussed in the essays for that prompt. So for example, the five most important words in the most frequently discussed topic for the military prompt we mentioned above are "man", "military", "service", "pay", and "war".

We also use the MALLET-generated topic model to tell us how much of each essay is spent discussing each of the 1,000 topics. The model might tell us, for example, that a particular essay written on the military prompt spends 35% of the time discussing the "man", "military", "service", "pay", and "war" topic and 65% of the time discussing a topic whose most important words are "fully", "count", "ordinary", "czech", and "day". Since the latter topic is discussed so much in the essay and does not appear to have much to do with the military prompt, this essay should probably get a bad prompt adherence score. We construct 1,000 features from this topic model, one for each topic. Each feature's value is obtained by using the topic model to tell us how much of the essay was spent discussing the feature's corresponding topic. From these features, our regressor should be able to learn which topics are important to a good prompt adherent essay.

6. Manually Annotated LDA Topics A weakness of the LDA topics feature type is that it may result in a regressor that has trouble distinguishing between an infrequent topic that is adherent to the prompt and one that just represents an irrelevant digression. This is because an infrequent topic may not appear in the training set often enough for the regressor to make this judgment. We introduce the manually annotated LDA topics feature type to address this problem.

In order to construct manually annotated LDA topic features, we first build 13 topic models, one for each prompt, just as described in the section on LDA topic features. Rather than requesting models of 1,000 topics, however, we request models of only 100 topics[2]. We then go through all 13 lists of 100 topics as represented by their top ten words, manually annotating each topic with a number from 0 to 5 representing how likely it is that the topic is adherent to the prompt. A topic labeled 5 is very likely to be related to the prompt, where a topic labeled 0 appears totally unrelated.

Using these annotations alongside the topic distribution for each essay that the topic models provide us, we construct ten features. The first five features encode the sum of the contributions to an essay of topics annotated with a number ≥ 1, the sum of the contributions to an essay of topics annotated with a number ≥ 2, and so on up to 5.

The next five features are similar to the last, with one feature taking on the sum of the contributions to an essay of topics annotated with the number 0, another feature taking on the sum of the

[2]We use 100 topics for each prompt in the manually annotated version of LDA features rather than the 1,000 topics we use in the regular version of LDA features because 1,300 topics are not too costly to annotate, but manually annotating 13,000 topics would take too much time.

contributions to an essay of topics annotated with the number 1, and so on up to 4. We do not include a feature for topics annotated with the number 5 because it would always have the same value as the feature for topics ≥ 5.

Features like these should give the regressor a better idea how much of an essay is composed of prompt-related arguments and discussion and how much of it is irrelevant to the prompt, even if some of the topics occurring in it are too infrequent to judge just from training data.

7. Predicted Thesis Clarity Errors In our previous work on essay thesis clarity scoring (Persing and Ng, 2013), we identified five classes of errors that detract from the clarity of an essay's thesis:
Confusing Phrasing. The thesis is phrased oddly, making it hard to understand the writer's point.
Incomplete Prompt Response. The thesis leaves some part of a multi-part prompt unaddressed.
Relevance to Prompt. The apparent thesis's weak relation to the prompt causes confusion.
Missing Details. The thesis leaves out an important detail needed to understand the writer's point.
Writer Position. The thesis describes a position on the topic without making it clear that this is the position the writer supports.

We hypothesize that these errors, though originally intended for thesis clarity scoring, could be useful for prompt adherence scoring as well. For instance, an essay that has a Relevance to Prompt error or an Incomplete Prompt Response error should intuitively receive a low prompt adherence score. For this reason, we introduce features based on these errors to our feature set for prompt adherence scoring[3].

While each of the essays in our data set was previously annotated with these thesis clarity errors, in a realistic setting a prompt adherence scoring system will not have access to these manual error labels. As a result, we first need to predict which of these errors is present in each essay. To do this, we train five maximum entropy classifiers for each prompt, one for each of the five thesis clarity errors, using MALLET's (McCallum, 2002) implementation of maximum entropy classification. Instances are presented to classifier for prompt p for error e in the following way. If a training essay is written in response to p, it will be used to generate a training instance whose label is 1 if e was annotated for it or 0 otherwise. Since error prediction and prompt adherence scoring are related problems, the features we associate with this instance are features $1-6$ which we have described earlier in this section. The classifier is then used to generate probabilities telling us how likely it is that each test essay has error e.

Then, when training our regressor for prompt adherence scoring, we add the following features to our instances. We add a binary feature indicating the presence or absence of each error. Or in the case of test essays, the feature takes on a real value from 0 to 1 indicating how likely the classifier thought it was that the essay had each of the errors. This results in five additional features, one for each error.

5 Evaluation

In this section, we evaluate our system for prompt adherence scoring. All the results we report are obtained via five-fold cross-validation experiments. In each experiment, we use $\frac{3}{5}$ of our labeled essays for model training, another $\frac{1}{5}$ for parameter tuning, and the final $\frac{1}{5}$ for testing.

5.1 Experimental Setup

5.1.1 Scoring Metrics

We employ four evaluation metrics. As we will see below, $S1$, $S2$, and $S3$ are *error* metrics, so lower scores imply better performance. In contrast, PC is a *correlation* metric, so higher correlation implies better performance.

The simplest metric, $S1$, measures the frequency at which a system predicts the wrong score out of the seven possible scores. Hence, a system that predicts the right score only 25% of the time would receive an $S1$ score of 0.75.

The $S2$ metric measures the average distance between a system's score and the actual score. This metric reflects the idea that a system that predicts scores close to the annotator-assigned scores should be preferred over a system whose predictions are further off, even if both systems estimate the correct score at the same frequency.

The $S3$ metric measures the average square of the distance between a system's score predictions and the annotator-assigned scores. The intuition behind this system is that not only should we prefer a system whose predictions are close to the annotator scores, but we should also prefer

[3]See our website at `http://www.hlt.utdallas.edu/~persingq/ICLE/` for the complete list of error annotations.

one whose predictions are not too frequently very far away from the annotator scores. These three scores are given by:

$$\frac{1}{N}\sum_{A_j \neq E'_j} 1, \quad \frac{1}{N}\sum_{i=1}^{N} |A_j - E_j|, \quad \frac{1}{N}\sum_{i=1}^{N} (A_j - E_j)^2$$

where A_j, E_j, and E'_j are the annotator assigned, system predicted, and rounded system predicted scores[4] respectively for essay j, and N is the number of essays.

The last metric, PC, computes Pearson's correlation coefficient between a system's predicted scores and the annotator-assigned scores. PC ranges from -1 to 1. A positive (negative) PC implies that the two sets of predictions are positively (negatively) correlated.

5.1.2 Parameter Tuning

As mentioned earlier, for each prompt p_i, we train a linear regressor r_i using LIBSVM with regularization parameter c_i. To optimize our system's performance on the three error measures described previously, we use held-out validation data to independently tune each of the c_i values[5]. Note that each of the c_i values can be tuned independently because a c_i value that is optimal for predicting scores for p_i essays with respect to any of the error performance measures is necessarily also the optimal c_i when measuring that error on essays from all prompts. However, this is not case with Pearson's correlation coefficient, as the PC value for essays from all 13 prompts cannot be simplified as a weighted sum of the PC values obtained on each individual prompt. In order to obtain an optimal result as measured by PC, we jointly tune the c_i parameters to optimize the PC value achieved by our system on the same held-out validation data. However, an exact solution to this optimization problem is computationally expensive, as there are too many (7^{13}) possible combinations of c values to exhaustively search. Consequently, we find a local maximum by employing the simulated annealing algorithm (Kirkpatrick et al., 1983), altering one c_i value at a time to optimize PC while holding the remaining parameters fixed.

[4]Since our regressor assigns each essay a real value rather than an actual valid score, it would be difficult to obtain a reasonable $S1$ score without rounding the system estimated score to one of the possible values. For that reason, we round the estimated score to the nearest of the seven scores the human annotators were permitted to assign (1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0) only when calculating $S1$. For other scoring metrics, we only round the predictions to 1.0 or 4.0 if they fall outside the 1.0−4.0 range.

[5]For parameter tuning, we employ the following values. c_i may be assigned any of the values 10^0 10^1, 10^2, 10^3, 10^4, 10^5, or 10^6.

System	$S1$	$S2$	$S3$	PC
Baseline	.517	.368	.234	.233
Our System	.488	.348	.197	.360

Table 4: Five-fold cross-validation results for prompt adherence scoring.

5.2 Results and Discussion

Five-fold cross-validation results on prompt adherence score prediction are shown in Table 4. On the first line, this table shows that our baseline system, which recall uses only various RI features, predicts the wrong score 51.7% of the time. Its predictions are off by an average of .368 points, and the average squared distance between its predicted score and the actual score is .234. In addition, its predicted scores and the actual scores have a Pearson correlation coefficient of 0.233.

The results from our system, which uses all seven feature types described in Section 4, are shown in row 2 of the table. Our system obtains $S1$, $S2$, $S3$, and PC scores of .488, .348, .197, and .360 respectively, yielding a significant improvement over the baseline with respect to $S2$, $S3$, and PC with $p < 0.05$, $p < 0.01$, and $p < 0.06$ respectively[6]. While our system yields improvements by all four measures, its improvement over the baseline $S1$ score is not significant. These results mean that the greatest improvements our system makes are that it ensures that our score predictions are not too often very far away from an essay's actual score, as making such predictions would tend to drive up $S3$, yielding a relative error reduction in $S3$ of 15.8%, and it also ensures a better correlation between predicted and actual scores, thus yielding the 16.6% improvement in PC.[7] It also gives more modest improvements in how frequently exactly the right score is predicted ($S1$) and is better at predicting scores closer to the actual scores ($S2$).

5.3 Feature Ablation

To gain insight into how much impact each of the feature types has on our system, we perform fea-

[6]All significance tests are paired t-tests.

[7]These numbers are calculated $\frac{B-O}{B-P}$ where B is the baseline system's score, O is our system's score, and P is a perfect score. Perfect scores for error measures and PC are 0 and 1 respectively.

ture ablation experiments in which we remove the feature types from our system one-by-one.

Results of the ablation experiments when performed using the four scoring metrics are shown in Table 5. The top line of each subtable shows what our system's score would be if we removed just one of the feature types from our system. So to see how our system performs by the $S1$ metric if we remove only predicted thesis clarity error features, we would look at the first row of results of Table 5(a) under the column headed by the number 7 since predicted thesis clarity errors are the seventh feature type introduced in Section 4. The number here tells us that our system's $S1$ score without this feature type is .502. Since Table 4 shows that when our system includes this feature type (along with all the other feature types), it obtains an $S1$ score of .488, this feature type's removal costs our system .014 $S1$ points, and thus its inclusion has a beneficial effect on the $S1$ score.

From row 1 of Table 5(a), we can see that removing feature 4 yields a system with the best $S1$ score in the presence of the other feature types in this row. For this reason, we permanently remove feature 4 from the system before we generate the results on line 2. Thus, we can see what happens when we remove both feature 4 and feature 5 by looking at the second entry in row 2. And since removing feature 6 harms performance least in the presence of row 2's other feature types, we permanently remove both 4 and 6 from our feature set when we generate the third row of results. We iteratively remove the feature type that yields a system with the best performance in this way until we get to the last line, where only one feature type is used to generate each result.

Since the feature type whose removal yields the best system is always the rightmost entry in a line, the order of column headings indicates the relative importance of the feature types, with the leftmost feature types being most important to performance and the rightmost feature types being least important in the presence of the other feature types. This being the case, it is interesting to note that while the relative importance of different feature types does not remain exactly the same if we measure performance in different ways, we can see that some feature types tend to be more important than others in a majority of the four scoring metrics. Features 2 (n-grams), 3 (thesis clarity keywords), and 6 (manually annotated LDA topics) tend to be the most important feature types, as they tend to be the last feature types removed in the ablation subtables. Features 1 (RI) and 5 (LDA topics) are of middling importance, with neither ever being removed first or last, and each tending to have a moderate effect on performance. Finally, while features 4 (prompt adherence keywords) and 7 (predicted thesis clarity errors) may by themselves provide useful information to our system, in the presence of the other feature types they tend to be the least important to performance as they are often the first feature types removed.

While there is a tendency for some feature types to always be important (or unimportant) regardless of which scoring metric is used to measure per-

(a) Results using the $S1$ metric

3	5	1	7	2	6	4
.527	.502	.512	.502	.511	.500	.488
.527	.502	.512	.501	.513	.500	
.525	.508	.505	.505	.504		
.513	.527	.520	.513			
.523	.520	.506				
.541	.527					

(b) Results using the $S2$ metric

2	6	3	1	4	5	7
.356	.350	.348	.350	.349	.348	.348
.351	.349	.348	.348	.348	.347	
.351	.349	.348	.348	.347		
.350	.349	.348	.348			
.358	.351	.349				
.362	.352					

(c) Results using the $S3$ metric

2	6	1	5	4	7	3
.221	.201	.197	.197	.197	.197	.196
.215	.201	.197	.196	.196	.196	
.212	.203	.199	.197	.196		
.212	.203	.199	.197			
.212	.203	.199				
.223	.204					

(d) Results using the PC metric

6	3	2	1	7	5	4
.326	.332	.303	.344	.348	.348	.361
.326	.332	.304	.343	.348	.348	
.324	.337	.292	.345	.352		
.322	.337	.297	.346			
.316	.321	.323				
.218	.325					

Table 5: Feature ablation results. In each subtable, the first row shows how our system would perform if each feature type was removed. We remove the least important feature type, and show in the next row how the adjusted system would perform without each remaining type. For brevity, a feature type is referred to by its feature number: (1) RI; (2) n-grams; (3) thesis clarity keywords; (4) prompt adherence keywords; (5) LDA topics; (6) manually annotated LDA topics; and (7) predicted thesis clarity errors.

	$S1$			$S2$			$S3$			PC		
Gold	.25	.50	.75	.25	.50	.75	.25	.50	.75	.25	.50	.75
2.0	3.35	3.56	3.79	3.40	3.52	3.73	3.06	3.37	3.64	3.06	3.37	3.64
2.5	3.43	3.63	3.80	3.25	3.52	3.79	3.24	3.45	3.67	3.24	3.46	3.73
3.0	3.64	3.78	3.85	3.56	3.70	3.90	3.52	3.65	3.74	3.52	3.66	3.79
3.5	3.73	3.81	3.88	3.63	3.78	3.90	3.59	3.70	3.81	3.60	3.74	3.85
4.0	3.76	3.84	3.88	3.70	3.83	3.90	3.63	3.75	3.84	3.66	3.78	3.88

Table 6: Regressor scores for our system.

formance, the relative importance of different feature types does not always remain consistent if we measure performance in different ways. For example, while we identified feature 3 (thesis clarity keywords) as one of the most important feature types generally due to its tendency to have a large beneficial impact on performance, when we are measuring performance using $S3$, it is the least useful feature type. Furthermore, its removal increases the $S3$ score by a small amount, meaning that its inclusion actually makes our system perform worse with respect to $S3$. Though feature 3 is an extreme example, all feature types fluctuate in importance, as we see when we compare their orders of removal among the four ablation subtables. Hence, it is important to know how performance is measured when building a system for scoring prompt adherence.

Feature 3 is not the only feature type whose removal sometimes has a beneficial impact on performance. As we can see in Table 5(b), the removal of features 4, 5, and 7 improves our system's $S2$ score by .001 points. The same effect occurs in Table 5(c) when we remove features 4, 7, and 3. These examples illustrate that under some scoring metrics, the inclusion of some feature types is actively harmful to performance. Fortunately, this effect does not occur in any other cases than the two listed above, as most feature types usually have a beneficial or at least neutral impact on our system's performance.

For those feature types whose effect on performance is neutral in the first lines of ablation results (feature 4 in $S1$, features 3, 5, and 7 in $S2$, and features 1, 4, 5, and 7 in $S3$), it is important to note that their neutrality does not mean that they are unimportant. It merely means that they do not improve performance in the presence of other feature types. We can see this is the case by noting that they are not all the least important feature types in their respective subtables as indicated by column order. For example, by the time feature 1 gets permanently removed in Table 5(c), its removal harms performance by .002 $S3$ points.

5.4 Analysis of Predicted Scores

To more closely examine the behavior of our system, in Table 6 we chart the distributions of scores it predicts for essays having each gold standard score. As an example of how to read this table, consider the number 3.06 appearing in row 2.0 in the .25 column of the $S3$ region. This means that 25% of the time, when our system with parameters tuned for optimizing $S3$ is presented with a test essay having a gold standard score of 2.0, it predicts that the essay has a score less than or equal to 3.06.

From this table, we see that our system has a strong bias toward predicting more frequent scores as there are no numbers less than 3.0 in the table, and about 93.7% of all essays have gold standard scores of 3.0 or above. Nevertheless, our system does not rely entirely on bias, as evidenced by the fact that each column in the table has a tendency for its scores to ascend as the gold standard score increases, implying that our system has some success at predicting lower scores for essays with lower gold standard prompt adherence scores.

Another interesting point to note about this table is that the difference in error weighting between the $S2$ and $S3$ scoring metrics appears to be having its desired effect, as every entry in the $S3$ subtable is less than its corresponding entry in the $S2$ subtable due to the greater penalty the $S3$ metric imposes for predictions that are very far away from the gold standard scores.

6 Conclusion

We proposed a feature-rich approach to the under-investigated problem of predicting essay-level prompt adherence scores on student essays. In an evaluation on 830 argumentative essays selected from the ICLE corpus, our system significantly outperformed a Random Indexing based baseline by several evaluation metrics. To stimulate further research on this task, we make all our annotations, including our prompt adherence scores, the LDA topic annotations, and the error annotations publicly available.

References

Yigal Attali and Jill Burstein. 2006. Automated essay scoring with E-rater v.2.0. *Journal of Technology, Learning, and Assessment*, 4(3).

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

Chih-Chung Chang and Chih-Jen Lin, 2001. *LIBSVM: A library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent smeantic analysis. *Journal of American Society of Information Science*, 41(6):391–407.

Sylviane Granger, Estelle Dagneaux, Fanny Meunier, and Magali Paquot. 2009. *International Corpus of Learner English (Version 2)*. Presses universitaires de Louvain.

Derrick Higgins and Jill Burstein. 2007. Sentence similarity measures for essay coherence. In *Proceedings of the 7th International Workshop on Computational Semantics*.

Derrick Higgins, Jill Burstein, Daniel Marcu, and Claudia Gentile. 2004. Evaluating multiple aspects of coherence in student essays. In *Human Language Technologies: The 2004 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 185–192.

Derrick Higgins, Jill Burstein, and Yigal Attali. 2006. Identifying off-topic student essays without topic-specific training data. *Natural Language Engineering*, 12(2):145–159.

David Jurgens and Keith Stevens. 2010. The S-Space package: An open source package for word space models. In *Proceedings of the ACL 2010 System Demonstrations*, pages 30–35.

Pentti Kanerva, Jan Kristoferson, and Anders Holst. 2000. Random indexing of text samples for latent semantic analysis. In *Proceedings of the 22nd Annual Conference of the Cognitive Science Society*, pages 103–106.

Scott Kirkpatrick, C. D. Gelatt, and Mario P. Vecchi. 1983. Optimization by simulated annealing. *Science*, 220(4598):671–680.

Thomas K. Landauer and Susan T. Dutnais. 1997. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, pages 211–240.

Thomas K. Landauer, Darrell Laham, and Peter W. Foltz. 2003. Automated scoring and annotation of essays with the Intelligent Essay Assessor™. In *Automated Essay Scoring: A Cross-Disciplinary Perspective*, pages 87–112. Lawrence Erlbaum Associates, Inc., Mahwah, NJ.

Annie Louis and Derrick Higgins. 2010. Off-topic essay detection using short prompt texts. In *Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 92–95.

Andrew Kachites McCallum. 2002. MALLET: A Machine Learning for Language Toolkit. `http://mallet.cs.umass.edu`.

Eleni Miltsakaki and Karen Kukich. 2004. Evaluation of text coherence for electronic essay scoring systems. *Natural Language Engineering*, 10(1):25–55.

Robert Parker, David Graf, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2009. *English Gigaword Fourth Edition*. Linguistic Data Consortium, Philadelphia.

Isaac Persing and Vincent Ng. 2013. Modeling thesis clarity in student essays. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 260–269.

Isaac Persing, Alan Davis, and Vincent Ng. 2010. Modeling organization in student essays. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 229–239.

Magnus Sahlgren. 2005. An introduction to random indexing. In *Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering*.

Mark D. Shermis and Jill C. Burstein. 2003. *Automated Essay Scoring: A Cross-Disciplinary Perspective*. Lawrence Erlbaum Associates, Inc., Mahwah, NJ.

Mark D. Shermis, Jill Burstein, Derrick Higgins, and Klaus Zechner. 2010. Automated essay scoring: Writing assessment and instruction. In *International Encyclopedia of Education (3rd edition)*. Elsevier, Oxford, UK.

Yiming Yang and Jan O. Pedersen. 1997. A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning*, pages 412–420.

ConnotationWordNet: Learning Connotation over the Word+Sense Network

Jun Seok Kang **Song Feng** **Leman Akoglu** **Yejin Choi**
Department of Computer Science
Stony Brook University
Stony Brook, NY 11794-4400
junkang, songfeng, leman, ychoi@cs.stonybrook.edu

Abstract

We introduce *ConnotationWordNet*, a connotation lexicon over the network of *words* in conjunction with *senses*. We formulate the lexicon induction problem as collective inference over pairwise-Markov Random Fields, and present a loopy belief propagation algorithm for inference. The key aspect of our method is that it is the *first unified* approach that assigns the polarity of *both* word- and sense-level connotations, exploiting the innate bipartite graph structure encoded in WordNet. We present comprehensive evaluation to demonstrate the quality and utility of the resulting lexicon in comparison to existing connotation and sentiment lexicons.

1 Introduction

We introduce *ConnotationWordNet*, a connotation lexicon over the network of *words* in conjunction with *senses*, as defined in WordNet. A connotation lexicon, as introduced first by Feng et al. (2011), aims to encompass subtle shades of sentiment a word may conjure, even for seemingly objective words such as *"sculpture"*, *"Ph.D."*, *"rosettes"*. Understanding the rich and complex layers of connotation remains to be a challenging task. As a starting point, we study a more feasible task of learning the *polarity* of connotation.

For non-polysemous words, which constitute a significant portion of English vocabulary, learning the *general* connotation at the *word-level* (rather than at the *sense-level*) would be a natural operational choice. However, for polysemous words, which correspond to most frequently used words, it would be an overly crude assumption that the same connotative polarity should be assigned for all senses of a given word. For example, consider *"abound"*, for which lexicographers of WordNet prescribe two different senses:

- (v) **abound**: (be abundant of plentiful; exist in large quantities)
- (v) **abound, burst, bristle**: (be in a state of movement or action) *"The room abounded with screaming children"*; *"The garden bristled with toddlers"*

For the first sense, which is the most commonly used sense for *"abound"*, the general overtone of the connotation would seem positive. That is, although one can use this sense in both positive and negative contexts, this sense of *"abound"* seems to collocate more often with items that are good to be abundant (e.g., *"resources"*), than unfortunate items being abundant (e.g., *"complaints"*).

However, as for the second sense, for which *"burst"* and *"bristle"* can be used interchangeably with respect to this particular sense,[1] the general overtone is slightly more negative with a touch of unpleasantness, or at least not as positive as that of the first sense. Especially if we look up the WordNet entry for *"bristle"*, there are noticeably more negatively connotative words involved in its gloss and examples.

This word sense issue has been a universal challenge for a range of Natural Language Processing applications, including sentiment analysis. Recent studies have shown that it is fruitful to tease out subjectivity and objectivity corresponding to different senses of the same word, in order to improve computational approaches to sentiment analysis (e.g. Pestian et al. (2012), Mihalcea et al. (2012) Balahur et al. (2014)). Encouraged by these recent successes, in this study, we investigate if we can attain similar gains if we model the connotative polarity of senses separately.

There is one potential practical issue we would like to point out in building a sense-level lexical resource, however. End-users of such a lexicon may not wish to deal with Word Sense Disam-

[1] Hence a *sense* in WordNet is defined by *synset (= synonym set)*, which is the set of words sharing the same sense.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1544–1554,
Baltimore, Maryland, USA, June 23-25 2014.

biguation (WSD), which is known to be often too noisy to be incorporated into the pipeline with respect to other NLP tasks. As a result, researchers often would need to aggregate labels across different senses to derive the word-level label. Although such aggregation is not entirely unreasonable, it does not seem to be the most optimal and principled way of integrating available resources.

Therefore, in this work, we present the first unified approach that learns *both* sense- and word-level connotations simultaneously. This way, end-users will have access to more accurate sense-level connotation labels if needed, while also having access to more general word-level connotation labels. We formulate the lexicon induction problem as collective inference over pairwise-Markov Random Fields (pairwise-MRF) and derive a loopy belief propagation algorithm for inference.

The key aspect of our approach is that we exploit the innate bipartite graph structure between words and senses encoded in WordNet. Although our approach seems conceptually natural, previous approaches, to our best knowledge, have not directly exploited these relations between words and senses for the purpose of deriving lexical knowledge over words and senses collectively. In addition, previous studies (for both sentiment and connotation lexicons) aimed to produce only either of the two aspects of the polarity: word-level or sense-level, while we address both.

Another contribution of our work is the introduction of loopy belief propagation (loopy-BP) as a lexicon induction algorithm. Loopy-BP in our study achieves statistically significantly better performance over the constraint optimization approaches previously explored. In addition, it runs much faster and it is considerably easier to implement. Last but not least, by using probabilistic representation of pairwise-MRF in conjunction with Loopy-BP as inference, the resulting solution has the natural interpretation as the intensity of connotation. This contrasts to approaches that seek discrete solutions such as Integer Linear Programming(Papadimitriou and Steiglitz, 1998).

ConnotationWordNet, the final outcome of our study, is a new lexical resource that has connotation labels over both words and senses following the structure of WordNet. The lexicon is publicly available at: `http://www.cs.sunysb.edu/~junkang/connotation_wordnet`.)

In what follows, we will first describe the network of words and senses (Section 2), then introduce the representation of the network structure as pairwise Markov Random Fields, and a loopy belief propagation algorithm as collective inference (Section 3). We then present comprehensive evaluation (Section 4 & 5 & 6), followed by related work (Section 7) and conclusion (Section 8).

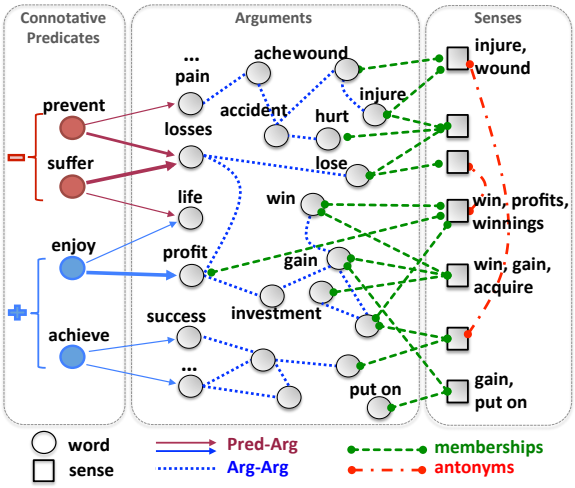


Figure 1: $G^{\text{WORD+SENSE}}$ with words and senses.

2 Network of Words and Senses

The connotation graph, called $G^{\text{WORD+SENSE}}$, is a heterogeneous graph with multiple types of nodes and edges. As shown in Figure 1, it contains two types of nodes; (i) lemmas (i.e., words, 115K) and (ii) synsets (63K), and four types of edges; (t_1) predicate-argument (179K), (t_2) argument-argument (144K), (t_3) argument-synset (126K), and (t_4) synset-synset (3.4K) edges.

The predicate-argument edges, first introduced by Feng et al. (2011), depict the selectional preference of connotative predicates (i.e., the polarity of a predicate indicates the polarity of its arguments) and encode their co-occurrence relations based on the Google Web 1T corpus. The argument-argument edges are based on the distributional similarities among the arguments. The argument-synset edges capture the synonymy between argument nodes through the corresponding synsets. Finally, the synset-synset edges depict the antonym relations between synset pairs.

In general, our graph construction is similar to that of Feng et al. (2013), but there are a few important differences. Most notably, we model both words and synsets explicitly, and exploit the membership relations between words and senses. We expect that edges between words and senses will encourage *senses that belong to the same word* to

receive the same connotation label. Conversely, we expect that these edges will also encourage *words that belong to the same sense* (i.e., synset definition) to receive the same connotation label.

Another benefit of our approach is that for various WordNet relations (e.g., antonym relations), which are defined over synsets (not over words), we can add edges directly between corresponding synsets, rather than projecting (i.e., approximating) those relations over words. Note that the latter, which has been employed by several previous studies (e.g., Kamps et al. (2004), Takamura et al. (2005), Andreevskaia and Bergler (2006), Su and Markert (2009), Lu et al. (2011), Kaji and Kitsuregawa (2007), Feng et al. (2013)), could be a source of noise, as one needs to assume that the semantic relation between a pair of synsets transfers over the pair of words corresponding to that pair of synsets. For polysemous words, this assumption may be overly strong.

3 Pairwise Markov Random Fields and Loopy Belief Propagation

We formulate the task of learning sense- and word-level connotation lexicon as a graph-based classification task (Sen et al., 2008). More formally, we denote the connotation graph $\text{G}^{\text{WORD+SENSE}}$ by $G = (V, E)$, in which a total of n word and synset nodes $V = \{v_1, \ldots, v_n\}$ are connected with typed edges $e(v_i, v_j, t) \in E$, where edge types $t \in \{pred\text{-}arg, arg\text{-}arg, syn\text{-}arg, syn\text{-}syn\}$ depict the four edge types as described in Section 2. A neighborhood function $\mathcal{N}$, where $\mathcal{N}_v = \{u|\ e(u, v) \in E\} \subseteq V$, describes the underlying network structure.

In our collective classification formulation, each node in V is represented as a random variable that takes a value from an appropriate class label domain; in our case, $\mathcal{L} = \{+, -\}$ for positive and negative connotation. In this classification task, we denote by $\mathcal{Y}$ the nodes the labels of which need to be assigned, and let y_i refer to Y_i's label.

3.1 Pairwise Markov Random Fields

We next define our objective function. We propose to use an objective formulation that utilizes pairwise Markov Random Fields (MRFs) (Kindermann and Snell, 1980), which we adapt to our problem setting. MRFs are a class of probabilistic graphical models that are suited for solving inference problems in networked data. An MRF consists of an undirected graph where each node can be in any of a finite number of states (i.e., class labels). The state of a node is assumed to be dependent on each of its neighbors and independent of other nodes in the graph.[2] In pairwise MRFs, the joint probability of the graph can be written as a product of pairwise factors, parameterized over the edges. These factors are referred to as clique potentials in general MRFs, which are essentially functions that collectively determine the graph's joint probability.

Specifically, let $G = (V, E)$ denote a network of random variables, where V consists of the unobserved variables $\mathcal{Y}$ that need to be assigned values from label set $\mathcal{L}$. Let Ψ denote a set of clique potentials that consists of two types of factors:

- For each $Y_i \in \mathcal{Y}$, $\psi_i \in \Psi$ is a *prior* mapping $\psi_i : \mathcal{L} \rightarrow \mathbb{R}_{\geq 0}$, where $\mathbb{R}_{\geq 0}$ denotes non-negative real numbers.
- For each $e(Y_i, Y_j, t) \in E$, $\psi_{ij}^t \in \Psi$ is a *compatibility* mapping $\psi_{ij}^t : \mathcal{L} \times \mathcal{L} \rightarrow \mathbb{R}_{\geq 0}$.

Objective formulation Given an assignment $\mathbf{y}$ to all the unobserved variables $\mathcal{Y}$ and $\mathbf{x}$ to observed ones $\mathcal{X}$ (variables with known labels, if any), our objective function is associated with the following joint probability distribution

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{Y_i \in \mathcal{Y}} \psi_i(y_i) \prod_{e(Y_i, Y_j, t) \in E} \psi_{ij}^t(y_i, y_j) \quad (1)$$

where $Z(\mathbf{x})$ is the normalization function. Our goal is then to infer the maximum likelihood assignment of states (i.e., labels) to unobserved variables (i.e., nodes) that will maximize Equation (1).

Problem Definition Having introduced our graph-based classification task and objective formulation, we define our problem more formally.

Given
- a connotation graph $G = (V, E)$ of words and synsets connected with *typed* edges,
- *prior* knowledge (i.e., probabilities) of (some or all) nodes belonging to each class,
- *compatibility* of two nodes with a given pair of labels being connected to each other;

Classify the nodes $Y_i \in \mathcal{Y}$, into one of two classes; $\mathcal{L} = \{+, -\}$, such that the class assignments y_i maximize our objective in Equation (1).

We can further *rank* the network objects by the probability of their connotation polarity.

[2] This assumption yields a *pairwise* Markov Random Field (MRF); a special case of general MRFs (Yedidia et al., 2003).

3.2 Loopy Belief Propagation

Finding the best assignments to unobserved variables in our objective function is the inference problem. The brute force approach through enumeration of all possible assignments is exponential and thus intractable. In general, exact inference is known to be NP-hard and there is no known algorithm which can be theoretically shown to solve the inference problem for general MRFs. Therefore in this work, we employ a computationally tractable (in fact linearly scalable with network size) approximate inference algorithm called Loopy Belief Propagation (LBP) (Yedidia et al., 2003), which we extend to handle typed graphs like our connotation graph.

Our inference algorithm is based on iterative message passing and the core of it can be concisely expressed as the following two equations:

$$m_{i\to j}(y_j) = \alpha \sum_{y_i \in \mathcal{L}} \left(\psi_{ij}^t(y_i, y_j)\ \psi_i(y_i) \prod_{Y_k \in \mathcal{N}_i \cap \mathcal{Y} \setminus Y_j} m_{k\to i}(y_i) \right), \quad \forall y_j \in \mathcal{L} \tag{2}$$

$$b_i(y_i) = \beta\ \psi_i(y_i) \prod_{Y_j \in \mathcal{N}_i \cap \mathcal{Y}} m_{j\to i}(y_i), \forall y_i \in \mathcal{L} \tag{3}$$

A message $m_{i\to j}$ is sent from node i to node j and captures the belief of i about j, which is the probability distribution over the labels of j; i.e. what i "thinks" j's label is, given the current label of i and the *type* of the edge that connects i and j. Beliefs refer to marginal probability distributions of nodes over labels; for example $b_i(y_i)$ denotes the *belief* of node i having label y_i. α and β are the normalization constants, which respectively ensure that each message and each set of marginal probabilities sum to 1. At every iteration, each node computes its belief based on messages received from its neighbors, and uses the compatibility mapping to transform its belief into messages for its neighbors. The key idea is that after enough iterations of message passes between the nodes, the "conversations" are likely to come to a consensus, which determines the marginal probabilities of all the unknown variables.

The pseudo-code of our method is given in Algorithm 1. It first initializes all messages to 1 and *prior*s to unbiased (i.e., equal) probabilities for all nodes except the seed nodes for which the sentiment is known (lines 3-9). It then proceeds by making each $Y_i \in \mathcal{Y}$ communicate messages

Algorithm 1: CONNOTATION INFERENCE

1 **Input:** Connotation graph G=(V, E), prior potentials ψ_s for seed words $s \in S$, and compatibility potentials ψ_{ij}^t
2 **Output:** Connotation label probabilities for each node $i \in V \setminus P$
3 **foreach** $e(Y_i, Y_j, t) \in E$ **do** // *initialize msg.s*
4 **foreach** $y_j \in \mathcal{L}$ **do**
5 $m_{i\to j}(y_j) \leftarrow 1$
6 **foreach** $i \in V$ **do** // *initialize priors*
7 **foreach** $y_j \in \mathcal{L}$ **do**
8 **if** $i \in S$ **then** $\phi_i(y_j) \leftarrow \psi_i(y_j)$ **else** $\phi_i(y_j) \leftarrow 1/|\mathcal{L}|$
9 **repeat** // *iterative message passing*
10 **foreach** $e(Y_i, Y_j, t) \in E$, $Y_j \in \mathcal{Y}^{V\setminus S}$ **do**
11 **foreach** $y_j \in \mathcal{L}$ **do**
12 Use Equation (2)
13 **until** all messages stop changing
14 **foreach** $Y_i \in \mathcal{Y}^{V\setminus S}$ **do** // *compute final beliefs*
15 **foreach** $y_i \in \mathcal{L}$ **do**
16 Use Equation (3)

with their neighbors in an iterative fashion until the messages stabilize (lines 10-14), i.e. convergence is reached.[3] At convergence, we calculate the marginal probabilities, that is of assigning Y_i with label y_i, by computing the final beliefs $b_i(y_i)$ (lines 15-17). We use these maximum likelihood probabilities for label assignment; for each node i, we assign the label $\mathcal{L}_i \leftarrow \max_{y_i} b_i(y_i)$.

To completely define our algorithm, we need to instantiate the potentials Ψ, in particular the priors and the compatibilities, which we discuss next.

Priors The *prior* beliefs ψ_i of nodes can be suitably initialized if there is any prior knowledge for their connotation sentiment (e.g., `enjoy` is positive, `suffer` is negative). As such, our method is flexible to integrate available side information. In case there is no prior knowledge available, each node is initialized equally likely to have any of the possible labels, i.e., $\frac{1}{|\mathcal{L}|}$ as in Algorithm 1 (line 9).

Compatibilities The *compatibility* potentials can be thought of as matrices, with entries

[3] Although convergence is not theoretically guaranteed, in practice LBP converges to beliefs within a small threshold of change (e.g., 10^{-6}) fairly quickly with accurate results (Pandit et al., 2007; McGlohon et al., 2009; Akoglu et al., 2013).

$\psi_{ij}^t(y_i, y_j)$ that give the likelihood of a node having label y_i, given that it has a neighbor with label y_j to which it is connected through a type t edge. A key difference of our method from earlier models is that we use clique potentials that differ for edge types, since the connotation graph is heterogeneous. This is exactly because the compatibility of class labels of two adjacent nodes depends on the type of the edge connecting them: e.g., $+ \xrightarrow{syn\text{-}arg} +$ is highly compatible, whereas $+ \xrightarrow{syn\text{-}syn} +$ is unlikely; as $syn\text{-}arg$ edges capture synonymy; i.e., words-sense memberships, while $syn\text{-}syn$ edges depict antonym relations.

A sample instantiation of the compatibilities is shown in Table 1. Notice that the potentials for $pred\text{-}arg$, $arg\text{-}arg$, and $syn\text{-}arg$ capture homophily, i.e., nodes with the same label are likely to connect to each other through these types of edges.[4] On the other hand, $syn\text{-}syn$ edges connect nodes that are antonyms of each other, and thus the compatibilities capture the reverse relationship among their labels.

Table 1: Instantiation of compatibility potentials. Entry $\psi_{ij}^t(y_i, y_j)$ is the compatibility of a node with label y_i having a neighbor labeled y_j, given the edge between i and j is type t, for small ϵ.

t: t_1	**A**	
P	+	−
+	1-ϵ	ϵ
−	ϵ	1-ϵ

(t_1) $pred\text{-}arg$

t: t_2	**A**	
A	+	−
+	1-2ϵ	2ϵ
−	2ϵ	1-2ϵ

(t_2) $arg\text{-}arg$

t: t_3	**A**	
S	+	−
+	1-ϵ	ϵ
−	ϵ	1-ϵ

(t_3) $syn\text{-}arg$
(synonym relations)

t: t_4	**S**	
S	+	−
+	ϵ	1-ϵ
−	1-ϵ	ϵ

(t_4) $syn\text{-}syn$
(antonym relations)

Complexity analysis Most demanding component of Algorithm 1 is the iterative message passing over the edges (lines 10-14), with time complexity $O(ml^2r)$, where $m = |E|$ is the number of edges in the connotation graph, $l = |\mathcal{L}|$, the classes, and r, the iterations until convergence. Often, l is quite small (in our case, $l = 2$) and $r \ll m$. Thus running time grows linearly with the number of edges and is scalable to large datasets.

[4] $arg\text{-}arg$ edges are based on co-occurrence (see Section 2), which does not carry as strong indication of the same connotation as e.g., synonymy. Thus, we enforce less homophily for nodes connected through edges of $arg\text{-}arg$ type.

4 Evaluation I: Agreement with Sentiment Lexicons

ConnotationWordNet is expected to be the superset of a sentiment lexicon, as it is highly likely for any word with positive/negative sentiment to carry connotation of the same polarity. Thus, we use two conventional sentiment lexicons, General Inquirer (GENINQ) (Stone et al., 1966) and MPQA (Wilson et al., 2005b), as surrogates to measure the performance of our inference algorithm.

4.1 Variants of Graph Construction

The construction of the connotation graph, denoted by $G^{WORD+SENSE}$, which includes words and synsets, has been described in Section 2. In addition to this graph, we tried several other graph constructions, the first three of which have previously been used in (Feng et al., 2013). We briefly describe these graphs below, and compare performance on all the graphs in the proceeding.

G^{WORD} W/ PRED-ARG: This is a (bipartite) subgraph of $G^{WORD+SENSE}$, which only includes the connotative predicates and their arguments. As such, it contains only type t_1 edges. The edges between the predicates and the arguments can be weighted by their Point-wise Mutual Information (PMI)[5] based on the Google Web 1T corpus.

G^{WORD} W/ OVERLAY: The second graph is also a proper subgraph of $G^{WORD+SENSE}$, which includes the predicates and all the argument words. Predicate words are connected to their arguments as before. In addition, argument pairs (a_1, a_2) are connected if they occurred together in the "a_1 and a_2" or "a_2 and a_1" coordination (Hatzivassiloglou and McKeown, 1997; Pickering and Branigan, 1998). This graph contains both type t_1 and t_2 edges. The edges can also be weighted based on the distributional similarities of the word pairs.

G^{WORD}: The third graph is a super-graph of G^{WORD} W/ OVERLAY, with additional edges, where argument pairs in synonym and antonym relation are connected to each other. Note that unlike the connotation graph $G^{WORD+SENSE}$, it does *not* contain any synset nodes. Rather, the words that are synonyms or antonyms of each other are directly linked in the graph. As such, this graph contains all edge types t_1 through t_4.

[5] PMI scores are widely used in previous studies to measure association between words (e.g., (Church and Hanks, 1990), (Turney, 2001), (Newman et al., 2009)).

G$^{\text{WORD+SENSE}}$ W/ SYNSIM: This is a super-graph of our original G$^{\text{WORD+SENSE}}$ graph; that is, it has all the predicate, arguments, and synset nodes, as well as the four types of edges between them. In addition, we add edges of a fifth type t_5 between the synset nodes to capture their similarity. To define similarity, we use the glossary definitions of the synsets and derive three different scores. Each score utilizes the `count`(s_1, s_2) of overlapping nouns, verbs, and adjectives/adverbs among the glosses of the two synsets s_1 and s_2.

G$^{\text{WORD+SENSE}}$ W/ SYNSIM1: We discard edges with `count` less than 3. The weighted version has the `counts` normalized between 0 and 1.

G$^{\text{WORD+SENSE}}$ W/ SYNSIM2: We normalize the `counts` by the length of the gloss (the avg of two lengths), that is, p = `count / avg(len_gloss(`s_1`), len_gloss(`s_2`))` and discard edges with $p < 0.5$. The weighted version contains p values as edge weights.

G$^{\text{WORD+SENSE}}$ W/ SYNSIM3: To further sparsify the graph we discard edges with $p < 0.6$. To weigh the edges, we use the cosine similarity between the gloss vectors of the synsets based on the TF-IDF values of the words the glosses contain.

Note that the connotation inference algorithm, as given in Algorithm 1, remains exactly the same for all the graphs described above. The only difference is the set of parameters used; while G$^{\text{WORD}}$ W/ PRED-ARG and G$^{\text{WORD}}$ W/ OVERLAY contain one and two edge types, respectively and only use compatibilities (t_1) and (t_2), G$^{\text{WORD}}$ uses all four as given in Table 1. The G$^{\text{WORD+SENSE}}$ W/ SYNSIM graphs use an additional compatibility matrix for the synset similarity edges of type t_5, which is the same as the one used for t_1, i.e., similar synsets are likely to have the same connotation label. This flexibility is one of the key advantages of our algorithm as new types of nodes and edges can be added to the graph seamlessly.

4.2 Sentiment-Lexicon based Performance

In this section, we first compare the performance of our connotation graph G$^{\text{WORD+SENSE}}$ to graphs that do not include synset nodes but only words. Then we analyze the performance when the additional synset similarity edges are added. First, we briefly describe our performance measures.

The sentiment lexicons we use as gold standard are small, compared to the size (i.e., number of words) our graphs contain. Thus, we first find the `overlap` between each graph and a sentiment lexicon. Note that the `overlap` size may be smaller than the `lexicon size`, as some sentiment words may be missing from our graphs. Then, we calculate the number of `correct` label assignments. As such, precision is defined as (`correct` / `overlap`), and recall as (`correct` / `lexicon size`). Finally, F1-score is their harmonic mean and reflects the overall accuracy.

	GENINQ			MPQA
	P	**R**	**F**	**F**
Variations of G$^{\text{WORD}}$				
W/ PRED-ARG	88.0	67.6	76.5	57.3
W/ PRED-ARG-W	84.9	68.9	76.1	57.8
W/ OVERLAY	87.8	70.4	78.1	58.4
W/ OVERLAY-W	82.2	67.7	74.2	54.2
G$^{\text{WORD}}$	88.5	83.1	85.7	69.7
G$^{\text{WORD}}$-W	75.5	71.5	73.4	53.2
Variations of G$^{\text{WORD+SENSE}}$				
G$^{\text{WORD+SENSE}}$	**88.8**	**84.1**	**86.4**	**70.0**
G$^{\text{WORD+SENSE}}$-W	76.8	73.0	74.9	54.6
W/ SYNSIM1	87.2	83.3	85.2	67.9
W/ SYNSIM2	83.9	80.8	82.3	65.1
W/ SYNSIM3	86.5	83.2	84.8	67.8
W/ SYNSIM1-W	88.0	**84.3**	86.1	69.2
W/ SYNSIM2-W	86.4	83.7	85.0	68.5
W/ SYNSIM3-W	86.7	83.4	85.0	68.2

Table 2: Connotation inference performance on various graphs. '-W' indicates weighted versions (see §4.1). **P**: precision, **R**: recall, **F**: F1-score (%).

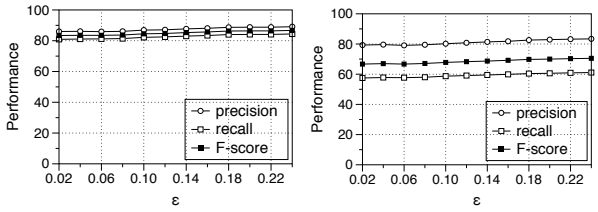
As shown in Table 2 (top), we first observe that including the synonym and antonym relations in the graph, as with G$^{\text{WORD}}$ and G$^{\text{WORD+SENSE}}$, improve the performance significantly, almost by an order of magnitude, over graphs G$^{\text{WORD}}$ W/ PRED-ARG and G$^{\text{WORD}}$ W/ OVERLAY that do not contain those relation types. Furthermore, we notice that the performances on the G$^{\text{WORD+SENSE}}$ graph are better than those on the word-only graphs. This shows that including the synset nodes explicitly in the graph structure is beneficial. What is more, it gives us a means to obtain connotation labels for the synsets themselves, which we use in the evaluations in the next sections. Finally, we note that using the unweighted versions of the graphs provide relatively more robust performance, potentially due to noise in the relative edge weights.

Next we analyze the performance when the new edges between synsets are introduced, as given in Table 2 (bottom). We observe that connecting the synset nodes by their gloss-similarity (at least in the ways we tried) does not yield better performance than on our original G$^{\text{WORD+SENSE}}$ graph. Different from earlier, the weighted versions of the similarity based graphs provide better perfor-

mance than their unweighted counterparts. This suggests that glossary similarity would be a more robust means to correlate nodes; we leave it as future work to explore this direction for predicate-argument and argument-argument relations.

4.3 Parameter Sensitivity

Our belief propagation based connotation sentiment inference algorithm has one user-specified parameter ϵ (see Table 1). To study the sensitivity of its performance to the choice of ϵ, we reran our experiments for $\epsilon = \{0.02, 0.04, \ldots, 0.24\}$[6] and report the accuracy results on our $G^{\text{WORD+SENSE}}$ in Figure 2 for the two lexicons. The results indicate that the performances remain quite stable across a wide range of the parameter choice.



(a) GENINQ EVAL (b) MPQA EVAL

Figure 2: Performance is stable across various ϵ.

5 Evaluation II: Human Evaluation on ConnotationWordNet

In this section, we present the result of human evaluation we executed using Amazon Mechanical Turk (AMT). We collect two separate sets of labels: a set of labels at the word-level, and another set at the sense-level. We first describe the labeling process of sense-level connotation: We selected 350 polysemous words and one of their senses, and each Turker was asked to rate the connotative polarity of a given word (or of a given sense), from -5 to 5, 0 being the neutral.[7] For each word, we asked 5 Turkers to rate and we took the average of the 5 ratings as the connotative intensity score of the word. We labeled a word as *negative* if its intensity score is less than 0 and *positive* otherwise. For word-level labels we apply similar procedure as above.

[6]Note that for $\epsilon > 0.25$, compatibilities of ψ^{t_2} in Table 1 are reversed, hence the maximum of 0.24.

[7]Because senses in WordNet can be tricky to understand, care should be taken in designing the task so that the Turkers will focus only on the corresponding sense of a word. Therefore, we provided the part of speech tag, the WordNet gloss of the selected sense, and a few examples as given in WordNet. As an incentive, each Turker was rewarded $0.07 per hit which consists of 10 words to label.

Lexicon	Word-level	Sense-level
SentiWordNet	27.22	14.29
OpinionFinder	31.95	-
Feng2013	62.72	-
$G^{\text{WORD+SENSE}}(95\%)$	84.91	83.43
$G^{\text{WORD+SENSE}}(99\%)$	84.91	83.71
E-$G^{\text{WORD+SENSE}}(95\%)$	**86.98**	**86.29**
E-$G^{\text{WORD+SENSE}}(99\%)$	86.69	85.71

Table 3: Word-/Sense-level evaluation results

5.1 Word-Level Evaluation

We first evaluate the word-level assignment of connotation, as shown in Table 3. The agreement between the new lexicon and human judges varies between 84% and 86.98%. Sentiment lexicons such as SentiWordNet (Baccianella et al. (2010)) and OpinionFinder (Wilson et al. (2005a)) show low agreement rate with human, which is somewhat as expected: human judges in this study are labeling for subtle connotation, not for more explicit sentiment. OpinionFinder's low agreement rate was mainly due to the low hit rate of the words (successful look-up rate, 33.43%). Feng2013 is the lexicon presented in (Feng et al., 2013) and it showed a relatively higher 72.13% hit rate.

Note that belief propagation was run until 95% and 99% of the nodes were converged in their beliefs. In addition, the seed words with known connotation labels originally consist of 20 positive and 20 negative predicates. We also extended the seed set with the sentiment lexicon words and denote these runs with E- for 'Extended'.

5.2 Sense-Level Evaluation

We also examined the agreement rates on the sense-level. Since OpinionFinder and Feng2013 do not provide the polarity scores at the sense-level, we excluded them from this evaluation. Because sense-level polarity assignment is a harder (more subtle) task, the performance of all lexicons decreased to some degree in comparison to that of word-level evaluations.

5.3 Pair-wise Intensity Ranking

A notable goodness of our induction algorithm is that the outcome of the algorithm can be interpreted as an *intensity* of the corresponding connotation. But are these values meaningful? We answer this question in this section. We formulate a pair-wise ranking task as a binary decision task as follows: given a pair of words, we ask which one is more positive (or more negative) than the other. Since we collect human labels based on *scales*, we

Lexicon	Correct	Undecided
SentiWordNet	33.77	23.34
$G^{\text{WORD+SENSE}}$(95%)	74.83	0.58
$G^{\text{WORD+SENSE}}$(99%)	73.01	0.58
E-$G^{\text{WORD+SENSE}}$(95%)	73.84	1.16
E-$G^{\text{WORD+SENSE}}$(99%)	74.01	1.16

Table 4: Results of pair-wise intensity evaluation, for intensity difference threshold = 2.0

already have this information at hand. Because different human judges have different notion of scales however, subtle differences are more likely to be noisy. Therefore, we experiment with varying degrees of differences in their scales, as shown in Figure 3. Threshold values (ranging from 0.5 to 3.0) indicate the minimum differences in scales for any pair of words, for the pair to be included in the test set. As expected, we observe that the performance improves as we increase the threshold (as pairs get better separated). Within range [0.5, 1.5] (249 pairs examined), the accuracies are as high as 68.27%, which shows that even the subtle differences of the connotative intensities are relatively well reflected in the new lexicons.

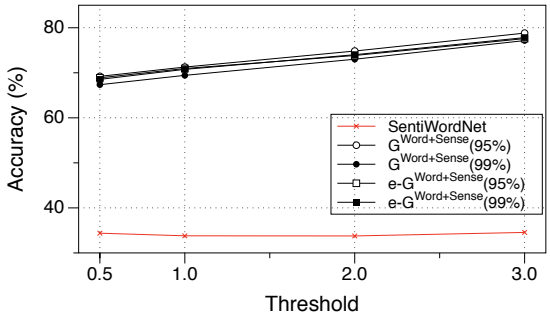


Figure 3: Trend of accuracy for pair-wise intensity evaluation over threshold

The results for pair-wise intensity evaluation (threshold=2.0, 1,208 pairs) are given in Table 4. Despite that intensity is generally a harder property to measure (than the coarser binary categorization of polarities), our connotation lexicons perform surprisingly well, reaching up to 74.83% accuracy. Further study on the incorrect cases reveals that SentiWordNet has many pair of words with the same polarity score (23.34%). Such cases seems to be due to the limited score patterns of SentiWordNet. The ratio of such cases are accounted as *Undecided* in Table 4.

6 Evaluation III: Sentiment Analysis using ConnotationWordNet

Finally, to show the utility of the resulting lexicon in the context of a concrete sentiment analysis task, we perform lexicon-based sentiment analysis. We experiment with SemEval dataset (Strapparava and Mihalcea, 2007) that includes the human labeled dataset for predicting whether a news headline is a *good news* or a *bad news*, which we expect to have a correlation with the use of *connotative* words that we focus on in this paper. The good/bad news are annotated with scores (ranging from -100 to 87). We construct several data sets by applying different thresholds on scores. For example, with the threshold set to 60, we discard the instances whose scores lie between -60 and 60. For comparison, we also test the connotation lexicon from (Feng et al., 2013) and the combined sentiment lexicon GENINQ+MPQA.

Note that there is a difference in how humans judge the orientation and the degree of connotation for a given word out of context, and how the use of such words in context can be perceived as *good/bad* news. In particular, we conjecture that humans may have a bias toward the use of positive words, which in turn requires calibration from the readers' minds (Pennebaker and Stone, 2003). That is, we might need to tone down the level of positiveness in order to correctly measure the actual intended positiveness of the message.

With this in mind, we tune the appropriate calibration from a small training data, by using 1 fold from N fold cross validation, and using the remaining $N-1$ folds as testing. We simply learn the mixture coefficient λ to scale the contribution of positive and negative connotation values. We tune this parameter λ[8] for other lexicons we compare against as well. Note that due to this parameter learning, we are able to report better performance for the connotation lexicon of (Feng et al., 2013) than what the authors have reported in their paper (labeled with *) in Table 5.

Table 5 shows the results for N=15, where the new lexicon consistently outperforms other competitive lexicons. In addition, Figure 4 shows that the performance does not change much based on the size of training data used for parameter tuning (N=$\{5, 10, 15, 20\}$).

7 Related Work

Several previous approaches explored the use of graph propagation for sentiment lexicon induction (Velikovich et al., 2010) and connotation lexicon

[8]What is reported is based on $\lambda \in \{20, 40, 60, 80\}$. More detailed parameter search does not change the results much.

Lexicon	SemEval Threshold			
	20	40	60	80
Instance Size	955	649	341	86
Feng2013	71.5	77.1	81.6	90.5
GENINQ+MPQA	72.8	77.2	80.4	86.7
$G^{\text{WORD+SENSE}}(95\%)$	74.5	**79.4**	86.5	**91.9**
$G^{\text{WORD+SENSE}}(99\%)$	**74.6**	**79.4**	**86.8**	**91.9**
E-$G^{\text{WORD+SENSE}}(95\%)$	72.5	76.8	82.3	87.2
E-$G^{\text{WORD+SENSE}}(99\%)$	72.6	76.9	82.5	87.2
Feng2013*	70.8	74.6	80.8	93.5
GENINQ+MPQA*	64.5	69.0	74.0	80.5

Table 5: SemEval evaluation results, for N=15

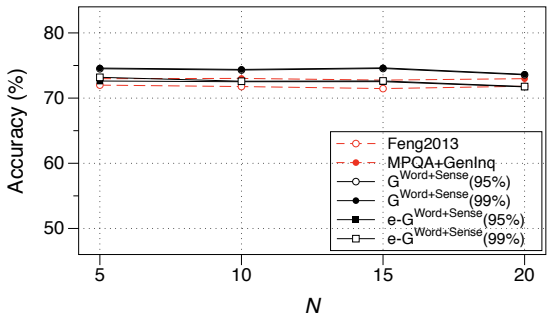


Figure 4: Trend of SemEval performance over N, the number of CV folds

induction (Feng et al., 2013). Our work introduces the use of loopy belief propagation over pairwise-MRF as an alternative solution to these tasks. At a high-level, both approaches share the general idea of propagating confidence or belief over the graph connectivity. The key difference, however, is that in our MRF representation, we can explicitly model various types of word-word, sense-sense and word-sense relations as edge potentials. In particular, we can naturally encode relations that encourage the same assignment (e.g., synonym) as well as the opposite assignment (e.g., antonym) of the polarity labels. Note that integration of the latter is not straightforward in the graph propagation framework.

There have been a number of previous studies that aim to construct a word-level sentiment lexicon (Wiebe et al., 2005; Qiu et al., 2009) and a sense-level sentiment lexicon (Esuli and Sebastiani, 2006). But none of these approaches considered to induce the polarity labels at both the word-level and sense-level. Although we focus on learning connotative polarity of words and senses in this paper, the same approach would be applicable to constructing a sentiment lexicon as well.

There have been recent studies that address word sense disambiguation issues for sentiment analysis. SentiWordNet (Esuli and Sebastiani, 2006) was the very first lexicon developed for sense-level labels of sentiment polarity. In recent years, Akkaya et al. (2009) report a successful empirical result where WSD helps improving sentiment analysis, while Wiebe and Mihalcea (2006) study the distinction between objectivity and subjectivity in each different sense of a word, and their empirical effects in the context of sentiment analysis. Our work shares the high-level spirit of accessing the sense-level polarity, while also deriving the word-level polarity.

In recent years, there has been a growing research interest in investigating more fine-grained aspects of lexical sentiment beyond positive and negative sentiment. For example, Mohammad and Turney (2010) study the affects words can *evoke* in people's minds, while Bollen et al. (2011) study various moods, e.g., "tension", "depression", beyond simple dichotomy of positive and negative sentiment. Our work, and some recent work by Feng et al. (2011) and Feng et al. (2013) share this spirit by targeting more subtle, nuanced sentiment even from those words that would be considered as objective in early studies of sentiment analysis.

8 Conclusion

We have introduced a novel formulation of lexicon induction operating over both words and senses, by exploiting the innate structure between the words and senses as encoded in WordNet. In addition, we introduce the use of loopy belief propagation over *pairwise*-Markov Random Fields as an effective lexicon induction algorithm. A notable strength of our approach is its expressiveness: various types of prior knowledge and lexical relations can be encoded as node potentials and edge potentials. In addition, it leads to a lexicon of better quality while also offering faster run-time and easiness of implementation. The resulting lexicon, called *ConnotationWordNet*, is the first lexicon that has polarity labels over both words and senses. *ConnotationWordNet* is publicly available for research and practical use.

Acknowledgments

This research was supported by the Army Research Office under Contract No. W911NF-14-1-0029, Stony Brook University Office of Vice President for Research, and gifts from Northrop Grumman Aerospace Systems and Google. We thank reviewers for many insightful comments and suggestions.

References

Cem Akkaya, Janyce Wiebe, and Rada Mihalcea. 2009. Subjectivity word sense disambiguation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 190–199. Association for Computational Linguistics.

Leman Akoglu, Rishi Chandy, and Christos Faloutsos. 2013. Opinion fraud detection in online reviews by network effects.

Alina Andreevskaia and Sabine Bergler. 2006. Mining wordnet for a fuzzy sentiment: Sentiment tag extraction from wordnet glosses. In *EACL*, pages 209–216.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 2200–2204.

Alexandra Balahur, Rada Mihalcea, and Andrés Montoyo. 2014. Computational approaches to subjectivity and sentiment analysis: Present and envisaged methods and applications. *Computer Speech & Language*, 28(1):1–6.

Johan Bollen, Huina Mao, and Alberto Pepe. 2011. Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena. In *ICWSM*.

K. W. Church and P. Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 1(16):22–29.

Andrea Esuli and Fabrizio Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *In Proceedings of the 5th Conference on Language Resources and Evaluation (LREC06*, pages 417–422.

Song Feng, Ritwik Bose, and Yejin Choi. 2011. Learning general connotation of words using graph-based algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1092–1103. Association for Computational Linguistics.

Song Feng, Jun Seok Kang, Polina Kuznetsova, and Yejin Choi. 2013. Connotation lexicon: A dash of sentiment beneath the surface meaning. In *The Association for Computer Linguistics*, pages 1774–1784.

Vasileios Hatzivassiloglou and Kathleen McKeown. 1997. Predicting the semantic orientation of adjectives. In *Proceedings of the Joint ACL/EACL Conference*, pages 174–181.

Nobuhiro Kaji and Masaru Kitsuregawa. 2007. Building lexicon for sentiment analysis from massive collection of html documents. In *EMNLP-CoNLL*, pages 1075–1083.

Jaap Kamps, MJ Marx, Robert J Mokken, and Maarten De Rijke. 2004. Using wordnet to measure semantic orientations of adjectives.

Ross Kindermann and J. L. Snell. 1980. *Markov Random Fields and Their Applications*.

Yue Lu, Malu Castellanos, Umeshwar Dayal, and ChengXiang Zhai. 2011. Automatic construction of a context-aware sentiment lexicon: an optimization approach. In *Proceedings of the 20th international conference on World wide web*, pages 347–356. ACM.

Mary McGlohon, Stephen Bay, Markus G. Anderle, David M. Steier, and Christos Faloutsos. 2009. Snare: a link analytic system for graph labeling and risk detection. In John F. Elder IV, Franoise Fogelman-Souli, Peter A. Flach, and Mohammed Zaki, editors, *KDD*, pages 1265–1274. ACM.

Rada Mihalcea, Carmen Banea, and Janyce Wiebe. 2012. Multilingual subjectivity and sentiment analysis. In *Tutorial Abstracts of ACL 2012*, pages 4–4. Association for Computational Linguistics.

Saif Mohammad and Peter Turney. 2010. Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 26–34, Los Angeles, CA, June. Association for Computational Linguistics.

David Newman, Sarvnaz Karimi, and Lawrence Cavedon. 2009. External evaluation of topic models. In *Australasian Document Computing Symposium*, pages 11–18, Sydney, December.

Shashank Pandit, Duen Horng Chau, Samuel Wang, and Christos Faloutsos. 2007. Netprobe: a fast and scalable system for fraud detection in online auction networks. In *WWW*, pages 201–210.

Christos H Papadimitriou and Kenneth Steiglitz. 1998. *Combinatorial optimization: algorithms and complexity*. Courier Dover Publications.

James W Pennebaker and Lori D Stone. 2003. Words of wisdom: language use over the life span. *Journal of personality and social psychology*, 85(2):291.

John P Pestian, Pawel Matykiewicz, Michelle Linn-Gust, Brett South, Ozlem Uzuner, Jan Wiebe, K Bretonnel Cohen, John Hurdle, Christopher Brew, et al. 2012. Sentiment analysis of suicide notes: A shared task. *Biomedical Informatics Insights*, 5(Suppl. 1):3.

Martin J. Pickering and Holly P. Branigan. 1998. The representation of verbs: Evidence from syntactic priming in language production. *Journal of Memory and Language*, 39:633–651.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2009. Expanding domain sentiment lexicon through double propagation. In *IJCAI*, volume 9, pages 1199–1204.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI Magazine*, 29(3):93–106.

Philip J. Stone, Dexter C. Dunphy, Marshall S. Smith, and Daniel M. Ogilvie. 1966. *The General Inquirer: A Computer Approach to Content Analysis*. MIT Press, Cambridge, MA.

Carlo Strapparava and Rada Mihalcea. 2007. Semeval-2007 task 14: Affective text. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 70–74. Association for Computational Linguistics.

Fangzhong Su and Katja Markert. 2009. Subjectivity recognition on word senses via semi-supervised mincuts. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1–9. Association for Computational Linguistics.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2005. Extracting semantic orientations of words using spin model. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 133–140. Association for Computational Linguistics.

Peter D. Turney. 2001. Mining the Web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the Twelfth European Conference on Machine Learning (ECML-01)*, pages 491–502, Freiburg, Germany.

Leonid Velikovich, Sasha Blair-Goldensohn, Kerry Hannan, and Ryan McDonald. 2010. The viability of web-derived polarity lexicons. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Janyce Wiebe and Rada Mihalcea. 2006. Word sense and subjectivity. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 1065–1072. Association for Computational Linguistics.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation (formerly Computers and the Humanities)*, 39(2/3):164–210.

Theresa Wilson, Paul Hoffmann, Swapna Somasundaran, Jason Kessler, Janyce Wiebe, Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005a. Opinionfinder: A system for subjectivity analysis. In *Proceedings of HLT/EMNLP on Interactive Demonstrations*, pages 34–35. Association for Computational Linguistics.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005b. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technologies Conference/Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP 2005)*, Vancouver, CA.

Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. 2003. Understanding belief propagation and its generalizations. In *Exploring AI in the new millennium*, pages 239–269.

Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification*

Duyu Tang[†], Furu Wei[‡] , Nan Yang[♮], Ming Zhou[‡], Ting Liu[†], Bing Qin[†]
[†]Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
[‡]Microsoft Research, Beijing, China
[♮]University of Science and Technology of China, Hefei, China
{dytang, tliu, qinb}@ir.hit.edu.cn
{fuwei, v-nayang, mingzhou}@microsoft.com

Abstract

We present a method that learns word embedding for Twitter sentiment classification in this paper. Most existing algorithms for learning continuous word representations typically only model the syntactic context of words but ignore the sentiment of text. This is problematic for sentiment analysis as they usually map words with similar syntactic context but opposite sentiment polarity, such as *good* and *bad*, to neighboring word vectors. We address this issue by learning sentiment-specific word embedding (**SSWE**), which encodes sentiment information in the continuous representation of words. Specifically, we develop three neural networks to effectively incorporate the supervision from sentiment polarity of text (e.g. sentences or tweets) in their loss functions. To obtain large scale training corpora, we learn the sentiment-specific word embedding from massive distant-supervised tweets collected by positive and negative emoticons. Experiments on applying SSWE to a benchmark Twitter sentiment classification dataset in SemEval 2013 show that (1) the SSWE feature performs comparably with hand-crafted features in the top-performed system; (2) the performance is further improved by concatenating SSWE with existing feature set.

1 Introduction

Twitter sentiment classification has attracted increasing research interest in recent years (Jiang et al., 2011; Hu et al., 2013). The objective is to classify the sentiment polarity of a tweet as positive, negative or neutral. The majority of existing approaches follow Pang et al. (2002) and employ machine learning algorithms to build classifiers from tweets with manually annotated sentiment polarity. Under this direction, most studies focus on designing effective features to obtain better classification performance. For example, Mohammad et al. (2013) build the top-performed system in the Twitter sentiment classification track of SemEval 2013 (Nakov et al., 2013), using diverse sentiment lexicons and a variety of hand-crafted features.

Feature engineering is important but labor-intensive. It is therefore desirable to discover explanatory factors from the data and make the learning algorithms less dependent on extensive feature engineering (Bengio, 2013). For the task of sentiment classification, an effective feature learning method is to compose the representation of a sentence (or document) from the representations of the words or phrases it contains (Socher et al., 2013b; Yessenalina and Cardie, 2011). Accordingly, it is a crucial step to learn the word representation (or word embedding), which is a dense, low-dimensional and real-valued vector for a word. Although existing word embedding learning algorithms (Collobert et al., 2011; Mikolov et al., 2013) are intuitive choices, they are not effective enough if directly used for sentiment classification. The most serious problem is that traditional methods typically model the syntactic context of words but ignore the sentiment information of text. As a result, words with opposite polarity, such as *good* and *bad*, are mapped into close vectors. It is meaningful for some tasks such as pos-tagging (Zheng et al., 2013) as the two words have similar usages and grammatical roles, but it becomes a disaster for sentiment analysis as they have the opposite sentiment polarity.

In this paper, we propose learning sentiment-specific word embedding (**SSWE**) for sentiment analysis. We encode the sentiment information in-

* This work was done when the first and third authors were visiting Microsoft Research Asia.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1555–1565,
Baltimore, Maryland, USA, June 23-25 2014.

to the continuous representation of words, so that it is able to separate *good* and *bad* to opposite ends of the spectrum. To this end, we extend the existing word embedding learning algorithm (Collobert et al., 2011) and develop three neural networks to effectively incorporate the supervision from sentiment polarity of text (e.g. sentences or tweets) in their loss functions. We learn the sentiment-specific word embedding from tweets, leveraging massive tweets with emoticons as distant-supervised corpora without any manual annotations. These automatically collected tweets contain noises so they cannot be directly used as gold training data to build sentiment classifiers, but they are effective enough to provide weakly supervised signals for training the sentiment-specific word embedding.

We apply SSWE as features in a supervised learning framework for Twitter sentiment classification, and evaluate it on the benchmark dataset in SemEval 2013. In the task of predicting positive/negative polarity of tweets, our method yields 84.89% in macro-F1 by only using SSWE as feature, which is comparable to the top-performed system based on hand-crafted features (84.70%). After concatenating the SSWE feature with existing feature set, we push the state-of-the-art to 86.58% in macro-F1. The quality of SSWE is also directly evaluated by measuring the word similarity in the embedding space for sentiment lexicons. In the accuracy of polarity consistency between each sentiment word and its top N closest words, SSWE outperforms existing word embedding learning algorithms.

The major contributions of the work presented in this paper are as follows.

- We develop three neural networks to learn sentiment-specific word embedding (SSWE) from massive distant-supervised tweets without any manual annotations;
- To our knowledge, this is the first work that exploits word embedding for Twitter sentiment classification. We report the results that the SSWE feature performs comparably with hand-crafted features in the top-performed system in SemEval 2013;
- We release the sentiment-specific word embedding learned from 10 million tweets, which can be adopted off-the-shell in other sentiment analysis tasks.

2 Related Work

In this section, we present a brief review of the related work from two perspectives, Twitter sentiment classification and learning continuous representations for sentiment classification.

2.1 Twitter Sentiment Classification

Twitter sentiment classification, which identifies the sentiment polarity of short, informal tweets, has attracted increasing research interest (Jiang et al., 2011; Hu et al., 2013) in recent years. Generally, the methods employed in Twitter sentiment classification follow traditional sentiment classification approaches. The lexicon-based approaches (Turney, 2002; Ding et al., 2008; Taboada et al., 2011; Thelwall et al., 2012) mostly use a dictionary of sentiment words with their associated sentiment polarity, and incorporate negation and intensification to compute the sentiment polarity for each sentence (or document).

The learning based methods for Twitter sentiment classification follow Pang et al. (2002)'s work, which treat sentiment classification of texts as a special case of text categorization issue. Many studies on Twitter sentiment classification (Pak and Paroubek, 2010; Davidov et al., 2010; Barbosa and Feng, 2010; Kouloumpis et al., 2011; Zhao et al., 2012) leverage massive noisy-labeled tweets selected by positive and negative emoticons as training set and build sentiment classifiers directly, which is called *distant supervision* (Go et al., 2009). Instead of directly using the distant-supervised data as training set, Liu et al. (2012) adopt the tweets with emoticons to smooth the language model and Hu et al. (2013) incorporate the emotional signals into an unsupervised learning framework for Twitter sentiment classification.

Many existing learning based methods on Twitter sentiment classification focus on feature engineering. The reason is that the performance of sentiment classifier being heavily dependent on the choice of feature representation of tweets. The most representative system is introduced by Mohammad et al. (2013), which is the state-of-the-art system (the top-performed system in SemEval 2013 Twitter Sentiment Classification Track) by implementing a number of hand-crafted features. Unlike the previous studies, we focus on learning discriminative features automatically from massive distant-supervised tweets.

2.2 Learning Continuous Representations for Sentiment Classification

Pang et al. (2002) pioneer this field by using bag-of-word representation, representing each word as a one-hot vector. It has the same length as the size of the vocabulary, and only one dimension is 1, with all others being 0. Under this assumption, many feature learning algorithms are proposed to obtain better classification performance (Pang and Lee, 2008; Liu, 2012; Feldman, 2013). However, the one-hot word representation cannot sufficiently capture the complex linguistic characteristics of words.

With the revival of interest in deep learning (Bengio et al., 2013), incorporating the continuous representation of a word as features has been proving effective in a variety of NLP tasks, such as parsing (Socher et al., 2013a), language modeling (Bengio et al., 2003; Mnih and Hinton, 2009) and NER (Turian et al., 2010). In the field of sentiment analysis, Bespalov et al. (2011; 2012) initialize the word embedding by Latent Semantic Analysis and further represent each document as the linear weighted of ngram vectors for sentiment classification. Yessenalina and Cardie (2011) model each word as a matrix and combine words using iterated matrix multiplication. Glorot et al. (2011) explore Stacked Denoising Autoencoders for domain adaptation in sentiment classification. Socher et al. propose Recursive Neural Network (RNN) (2011b), matrix-vector RNN (2012) and Recursive Neural Tensor Network (RNTN) (2013b) to learn the compositionality of phrases of any length based on the representation of each pair of children recursively. Hermann et al. (2013) present Combinatory Categorial Autoencoders to learn the compositionality of sentence, which marries the Combinatory Categorial Grammar with Recursive Autoencoder.

The representation of words heavily relies on the applications or tasks in which it is used (Labutov and Lipson, 2013). This paper focuses on learning sentiment-specific word embedding, which is tailored for sentiment analysis. Unlike Maas et al. (2011) that follow the probabilistic document model (Blei et al., 2003) and give an sentiment predictor function to each word, we develop neural networks and map each ngram to the sentiment polarity of sentence. Unlike Socher et al. (2011c) that utilize manually labeled texts to learn the meaning of phrase (or sentence) through compositionality, we focus on learning the meaning of word, namely word embedding, from massive distant-supervised tweets. Unlike Labutov and Lipson (2013) that produce task-specific embedding from an existing word embedding, we learn sentiment-specific word embedding from scratch.

3 Sentiment-Specific Word Embedding for Twitter Sentiment Classification

In this section, we present the details of learning sentiment-specific word embedding (**SSWE**) for Twitter sentiment classification. We propose incorporating the sentiment information of sentences to learn continuous representations for words and phrases. We extend the existing word embedding learning algorithm (Collobert et al., 2011) and develop three neural networks to learn SSWE. In the following sections, we introduce the traditional method before presenting the details of SSWE learning algorithms. We then describe the use of SSWE in a supervised learning framework for Twitter sentiment classification.

3.1 C&W Model

Collobert et al. (2011) introduce C&W model to learn word embedding based on the syntactic contexts of words. Given an ngram *"cat chills on a mat"*, C&W replaces the center word with a random word w^r and derives a **corrupted** ngram *"cat chills w^r a mat"*. The training objective is that the original ngram is expected to obtain a higher language model score than the corrupted ngram by a margin of 1. The ranking objective function can be optimized by a hinge loss,

$$loss_{cw}(t, t^r) = max(0, 1 - f^{cw}(t) + f^{cw}(t^r)) \quad (1)$$

where t is the original ngram, t^r is the corrupted ngram, $f^{cw}(\cdot)$ is a one-dimensional scalar representing the language model score of the input ngram. Figure 1(a) illustrates the neural architecture of C&W, which consists of four layers, namely $lookup \rightarrow linear \rightarrow hTanh \rightarrow linear$ (from bottom to top). The original and corrupted ngrams are treated as inputs of the feed-forward neural network, respectively. The output f^{cw} is the language model score of the input, which is calculated as given in Equation 2, where L is the lookup table of word embedding, w_1, w_2, b_1, b_2 are the parameters of linear layers.

$$f^{cw}(t) = w_2(a) + b_2 \quad (2)$$

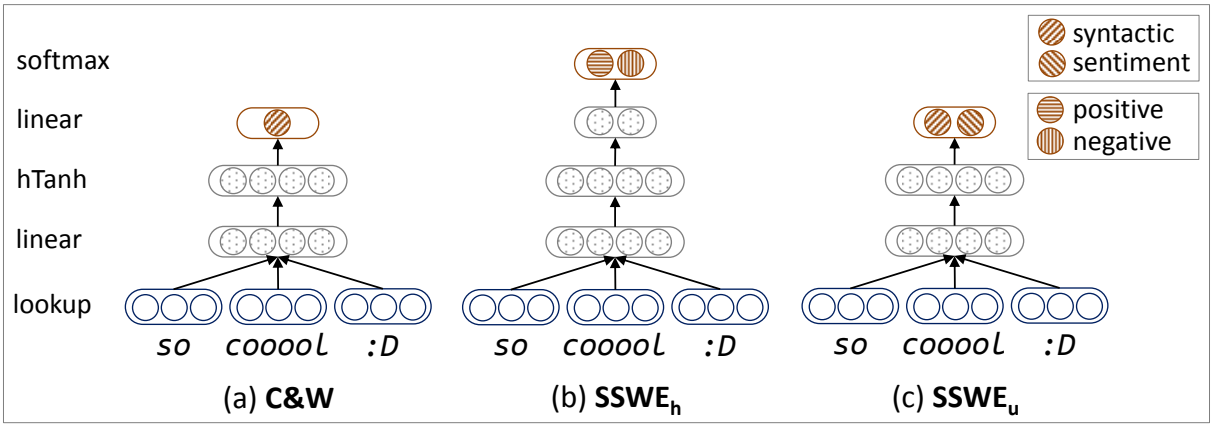


Figure 1: The traditional C&W model and our neural networks (SSWE$_h$ and SSWE$_u$) for learning sentiment-specific word embedding.

$$a = hTanh(w_1 L_t + b_1) \quad (3)$$

$$hTanh(x) = \begin{cases} -1 & \text{if } x < -1 \\ x & \text{if } -1 \leq x \leq 1 \\ 1 & \text{if } x > 1 \end{cases} \quad (4)$$

3.2 Sentiment-Specific Word Embedding

Following the traditional C&W model (Collobert et al., 2011), we incorporate the sentiment information into the neural network to learn sentiment-specific word embedding. We develop three neural networks with different strategies to integrate the sentiment information of tweets.

Basic Model 1 (SSWE$_h$). As an unsupervised approach, C&W model does not explicitly capture the sentiment information of texts. An intuitive solution to integrate the sentiment information is predicting the sentiment distribution of text based on input ngram. We do not utilize the entire sentence as input because the length of different sentences might be variant. We therefore slide the window of ngram across a sentence, and then predict the sentiment polarity based on each ngram with a shared neural network. In the neural network, the distributed representation of higher layer are interpreted as features describing the input. Thus, we utilize the continuous vector of top layer to predict the sentiment distribution of text.

Assuming there are K labels, we modify the dimension of top layer in C&W model as K and add a $softmax$ layer upon the top layer. The neural network (**SSWE$_h$**) is given in Figure 1(b). $Softmax$ layer is suitable for this scenario because its outputs are interpreted as conditional probabilities. Unlike C&W, SSWE$_h$ does not generate any corrupted ngram. Let $\boldsymbol{f}^g(t)$, where K denotes the number of sentiment polarity labels, be the gold K-dimensional multinomial distribution of input t and $\sum_k \boldsymbol{f}^g_k(t) = 1$. For positive/negative classification, the distribution is of the form [1,0] for positive and [0,1] for negative. The cross-entropy error of the $softmax$ layer is :

$$loss_h(t) = -\sum_{k=\{0,1\}} \boldsymbol{f}^g_k(t) \cdot log(\boldsymbol{f}^h_k(t)) \quad (5)$$

where $\boldsymbol{f}^g(t)$ is the gold sentiment distribution and $\boldsymbol{f}^h(t)$ is the predicted sentiment distribution.

Basic Model 2 (SSWE$_r$). SSWE$_h$ is trained by predicting the positive ngram as [1,0] and the negative ngram as [0,1]. However, the constraint of SSWE$_h$ is too strict. The distribution of [0.7,0.3] can also be interpreted as a positive label because the positive score is larger than the negative score. Similarly, the distribution of [0.2,0.8] indicates negative polarity. Based on the above observation, the hard constraints in SSWE$_h$ should be relaxed. If the sentiment polarity of a tweet is positive, the predicted positive score is expected to be larger than the predicted negative score, and the exact reverse if the tweet has negative polarity.

We model the relaxed constraint with a ranking objective function and borrow the bottom four layers from SSWE$_h$, namely $lookup \rightarrow linear \rightarrow hTanh \rightarrow linear$ in Figure 1(b), to build the relaxed neural network (**SSWE$_r$**). Compared with SSWE$_h$, the $softmax$ layer is removed because SSWE$_r$ does not require probabilistic interpretation. The hinge loss of SSWE$_r$ is modeled as de-

scribed below.

$$loss_r(t) = max(0, 1 - \delta_s(t)\boldsymbol{f}_0^r(t) + \delta_s(t)\boldsymbol{f}_1^r(t)\ ) \quad (6)$$

where $\boldsymbol{f}_0^r$ is the predicted positive score, $\boldsymbol{f}_1^r$ is the predicted negative score, $\delta_s(t)$ is an indicator function reflecting the sentiment polarity of a sentence,

$$\delta_s(t) = \begin{cases} 1 & \text{if } \boldsymbol{f}^g(t) = [1, 0] \\ -1 & \text{if } \boldsymbol{f}^g(t) = [0, 1] \end{cases} \quad (7)$$

Similar with SSWE$_h$, SSWE$_r$ also does not generate the corrupted ngram.

Unified Model (SSWE$_u$). The C&W model learns word embedding by modeling syntactic contexts of words but ignoring sentiment information. By contrast, SSWE$_h$ and SSWE$_r$ learn sentiment-specific word embedding by integrating the sentiment polarity of sentences but leaving out the syntactic contexts of words. We develop a unified model (**SSWE$_u$**) in this part, which captures the sentiment information of sentences as well as the syntactic contexts of words. SSWE$_u$ is illustrated in Figure 1(c).

Given an original (or corrupted) ngram and the sentiment polarity of a sentence as the input, SSWE$_u$ predicts a two-dimensional vector for each input ngram. The two scalars ($\boldsymbol{f}_0^u$, $\boldsymbol{f}_1^u$) stand for language model score and sentiment score of the input ngram, respectively. The training objectives of SSWE$_u$ are that (1) the original ngram should obtain a higher language model score $\boldsymbol{f}_0^u(t)$ than the corrupted ngram $\boldsymbol{f}_0^u(t^r)$, and (2) the sentiment score of original ngram $\boldsymbol{f}_1^u(t)$ should be more consistent with the gold polarity annotation of sentence than corrupted ngram $\boldsymbol{f}_1^u(t^r)$. The loss function of SSWE$_u$ is the linear combination of two hinge losses,

$$loss_u(t, t^r) = \alpha \cdot loss_{cw}(t, t^r) + (1 - \alpha) \cdot loss_{us}(t, t^r) \quad (8)$$

where $loss_{cw}(t, t^r)$ is the syntactic loss as given in Equation 1, $loss_{us}(t, t^r)$ is the sentiment loss as described in Equation 9. The hyper-parameter α weighs the two parts.

$$loss_{us}(t, t^r) = max(0, 1 - \delta_s(t)\boldsymbol{f}_1^u(t) + \delta_s(t)\boldsymbol{f}_1^u(t^r)\ ) \quad (9)$$

Model Training. We train sentiment-specific word embedding from massive distant-supervised tweets collected with positive and negative emoticons[1]. We crawl tweets from April 1st, 2013 to April 30th, 2013 with TwitterAPI. We tokenize each tweet with TwitterNLP (Gimpel et al., 2011), remove the *@user* and *URLs* of each tweet, and filter the tweets that are too short (< 7 words). Finally, we collect 10M tweets, selected by 5M tweets with positive emoticons and 5M tweets with negative emoticons.

We train SSWE$_h$, SSWE$_r$ and SSWE$_u$ by taking the derivative of the loss through back-propagation with respect to the whole set of parameters (Collobert et al., 2011), and use AdaGrad (Duchi et al., 2011) to update the parameters. We empirically set the window size as 3, the embedding length as 50, the length of hidden layer as 20 and the learning rate of AdaGrad as 0.1 for all baseline and our models. We learn embedding for unigrams, bigrams and trigrams separately with same neural network and same parameter setting. The contexts of unigram (bigram/trigram) are the surrounding unigrams (bigrams/trigrams), respectively.

3.3 Twitter Sentiment Classification

We apply sentiment-specific word embedding for Twitter sentiment classification under a supervised learning framework as in previous work (Pang et al., 2002). Instead of hand-crafting features, we incorporate the continuous representation of words and phrases as the feature of a tweet. The sentiment classifier is built from tweets with manually annotated sentiment polarity.

We explore min, $average$ and max convolutional layers (Collobert et al., 2011; Socher et al., 2011a), which have been used as simple and effective methods for compositionality learning in vector-based semantics (Mitchell and Lapata, 2010), to obtain the tweet representation. The result is the concatenation of vectors derived from different convolutional layers.

$$z(tw) = [z_{max}(tw), z_{min}(tw), z_{average}(tw)]$$

where $z(tw)$ is the representation of tweet tw and $z_x(tw)$ is the results of the convolutional layer $x \in \{min, max, average\}$. Each convolutional layer

[1] We use the emoticons selected by Hu et al. (2013). The positive emoticons are :) :) :-) :D =), and the negative emoticons are :(: (:-(.

z_x employs the embedding of unigrams, bigrams and trigrams separately and conducts the matrix-vector operation of x on the sequence represented by columns in each lookup table. The output of z_x is the concatenation of results obtained from different lookup tables.

$$z_x(tw) = [w_x\langle L_{uni}\rangle^{tw}, w_x\langle L_{bi}\rangle^{tw}, w_x\langle L_{tri}\rangle^{tw}]$$

where w_x is the convolutional function of z_x, $\langle L\rangle^{tw}$ is the concatenated column vectors of the words in the tweet. L_{uni}, L_{bi} and L_{tri} are the lookup tables of the unigram, bigram and trigram embedding, respectively.

4 Experiment

We conduct experiments to evaluate SSWE by incorporating it into a supervised learning framework for Twitter sentiment classification. We also directly evaluate the effectiveness of the SSWE by measuring the word similarity in the embedding space for sentiment lexicons.

4.1 Twitter Sentiment Classification

Experiment Setup and Datasets. We conduct experiments on the latest Twitter sentiment classification benchmark dataset in SemEval 2013 (Nakov et al., 2013). The training and development sets were completely in full to task participants. However, we were unable to download all the training and development sets because some tweets were deleted or not available due to modified authorization status. The test set is directly provided to the participants. The distribution of our dataset is given in Table 1. We train sentiment classifier with LibLinear (Fan et al., 2008) on the training set, tune parameter $-c$ on the dev set and evaluate on the test set. Evaluation metric is the Macro-F1 of positive and negative categories [2].

	Positive	Negative	Neutral	Total
Train	2,642	994	3,436	7,072
Dev	408	219	493	1,120
Test	1,570	601	1,639	3,810

Table 1: Statistics of the SemEval 2013 Twitter sentiment classification dataset.

Baseline Methods. We compare our method with the following sentiment classification algorithms:

(1) *DistSuper*: We use the 10 million tweets selected by positive and negative emoticons as training data, and build sentiment classifier with LibLinear and ngram features (Go et al., 2009).

(2) *SVM*: The ngram features and Support Vector Machine are widely used baseline methods to build sentiment classifiers (Pang et al., 2002). LibLinear is used to train the SVM classifier.

(3) *NBSVM*: NBSVM (Wang and Manning, 2012) is a state-of-the-art performer on many sentiment classification datasets, which trades-off between Naive Bayes and NB-enhanced SVM.

(4) *RAE*: Recursive Autoencoder (Socher et al., 2011c) has been proven effective in many sentiment analysis tasks by learning compositionality automatically. We run RAE with randomly initialized word embedding.

(5) *NRC*: NRC builds the top-performed system in SemEval 2013 Twitter sentiment classification track which incorporates diverse sentiment lexicons and many manually designed features. We re-implement this system because the codes are not publicly available [3]. *NRC-ngram* refers to the feature set of *NRC* leaving out ngram features.

Except for $DistSuper$, other baseline methods are conducted in a supervised manner. We do not compare with RNTN (Socher et al., 2013b) because we cannot efficiently train the RNTN model. The reason lies in that the tweets in our dataset do not have accurately parsed results or fine grained sentiment labels for phrases. Another reason is that the RNTN model trained on movie reviews cannot be directly applied on tweets due to the differences between domains (Blitzer et al., 2007).

Results and Analysis. Table 2 shows the macro-F1 of the baseline systems as well as the SSWE-based methods on positive/negative sentiment classification of tweets. Distant supervision is relatively weak because the noisy-labeled tweets are treated as the gold standard, which affects the performance of classifier. The results of bag-of-ngram (uni/bi/tri-gram) features are not satisfied because the one-hot word representation cannot capture the latent connections between words. NBSVM and RAE perform comparably and have

[2]We investigate 2-class Twitter sentiment classification (positive/negative) instead of 3-class Twitter sentiment classification (positive/negative/neutral) in SemEval2013.

[3]For 3-class sentiment classification in SemEval 2013, our re-implementation of NRC achieved 68.3%, 0.7% lower than NRC (69%) due to less training data.

Method	Macro-F1
DistSuper + unigram	61.74
DistSuper + uni/bi/tri-gram	63.84
SVM + unigram	74.50
SVM + uni/bi/tri-gram	75.06
NBSVM	75.28
RAE	75.12
NRC (**Top System** in SemEval)	**84.73**
NRC - ngram	84.17
$SSWE_u$	**84.98**
$SSWE_u$+NRC	**86.58**
$SSWE_u$+NRC-ngram	**86.48**

Table 2: Macro-F1 on positive/negative classification of tweets.

a big gap in comparison with the NRC and SSWE-based methods. The reason is that RAE and NBSVM learn the representation of tweets from the small-scale manually annotated training set, which cannot well capture the comprehensive linguistic phenomenons of words.

NRC implements a variety of features and reaches 84.73% in macro-F1, verifying the importance of a better feature representation for Twitter sentiment classification. We achieve 84.98% by using only $SSWE_u$ as features without borrowing any sentiment lexicons or hand-crafted rules. The results indicate that $SSWE_u$ automatically learns discriminative features from massive tweets and performs comparable with the state-of-the-art manually designed features. After concatenating $SSWE_u$ with the feature set of *NRC*, the performance is further improved to 86.58%. We also compare $SSWE_u$ with the ngram feature by integrating SSWE into *NRC-ngram*. The concatenated features *$SSWE_u$+NRC-ngram* (86.48%) outperform the original feature set of NRC (84.73%).

As a reference, we apply $SSWE_u$ on subjective classification of tweets, and obtain 72.17% in macro-F1 by using only $SSWE_u$ as feature. After combining $SSWE_u$ with the feature set of *NRC*, we improve NRC from 74.86% to 75.39% for subjective classification.

Comparision between Different Word Embedding. We compare sentiment-specific word embedding ($SSWE_h$, $SSWE_r$, $SSWE_u$) with baseline embedding learning algorithms by only using word embedding as features for Twitter sentiment classification. We use the embedding of unigrams, bigrams and trigrams in the experiment. The embeddings of C&W (Collobert et al., 2011), word2vec[4], WVSA (Maas et al., 2011) and our models are trained with the same dataset and same parameter setting. We compare with C&W and word2vec as they have been proved effective in many NLP tasks. The trade-off parameter of ReEmb (Labutov and Lipson, 2013) is tuned on the development set of SemEval 2013.

Table 3 shows the performance on the positive/negative classification of tweets[5]. ReEmb(C&W) and ReEmb(w2v) stand for the use of embeddings learned from 10 million distant-supervised tweets with C&W and word2vec, respectively. Each row of Table 3 represents a word embedding learning algorithm. Each column stands for a type of embedding used to compose features of tweets. The column *uni+bi* denotes the use of unigram and bigram embedding, and the column *uni+bi+tri* indicates the use of unigram, bigram and trigram embedding.

Embedding	unigram	uni+bi	uni+bi+tri
C&W	74.89	75.24	75.89
Word2vec	73.21	75.07	76.31
ReEmb(C&W)	75.87	–	–
ReEmb(w2v)	75.21	–	–
WVSA	77.04	–	–
$SSWE_h$	81.33	83.16	83.37
$SSWE_r$	80.45	81.52	82.60
$SSWE_u$	**83.70**	**84.70**	**84.98**

Table 3: Macro-F1 on positive/negative classification of tweets with different word embeddings.

From the first column of Table 3, we can see that the performance of C&W and word2vec are obviously lower than sentiment-specific word embeddings by only using unigram embedding as features. The reason is that C&W and word2vec do not explicitly exploit the sentiment information of the text, resulting in that the words with opposite polarity such as *good* and *bad* are mapped to close word vectors. When such word embeddings are fed as features to a Twitter sentiment classifier, the discriminative ability of sentiment words are weakened thus the classification performance is affected. Sentiment-specific word em-

[4]Available at https://code.google.com/p/word2vec/. We utilize the Skip-gram model because it performs better than CBOW in our experiments.

[5]MVSA and ReEmb are not suitable for learning bigram and trigram embedding because their sentiment predictor functions only utilize the unigram embedding.

beddings (SSWE$_h$, SSWE$_r$, SSWE$_u$) effectively distinguish words with opposite sentiment polarity and perform best in three settings. SSWE outperforms MVSA by exploiting more contextual information in the sentiment predictor function. SSWE outperforms ReEmb by leveraging more sentiment information from massive distant-supervised tweets. Among three sentiment-specific word embeddings, SSWE$_u$ captures more context information and yields best performance. SSWE$_h$ and SSWE$_r$ obtain comparative results.

From each row of Table 3, we can see that the bigram and trigram embeddings consistently improve the performance of Twitter sentiment classification. The underlying reason is that a phrase, which cannot be accurately represented by unigram embedding, is directly encoded into the n-gram embedding as an idiomatic unit. A typical case in sentiment analysis is that the composed phrase and multiword expression may have a different sentiment polarity than the individual words it contains, such as *not [bad]* and *[great] deal of* (the word in the bracket has different sentiment polarity with the ngram). A very recent study by Mikolov et al. (2013) also verified the effectiveness of phrase embedding for analogically reasoning phrases.

Effect of α in SSWE$_u$ We tune the hyper-parameter α of SSWE$_u$ on the development set by using unigram embedding as features. As given in Equation 8, α is the weighting score of syntactic loss of SSWE$_u$ and trades-off the syntactic and sentiment losses. SSWE$_u$ is trained from 10 million distant-supervised tweets.

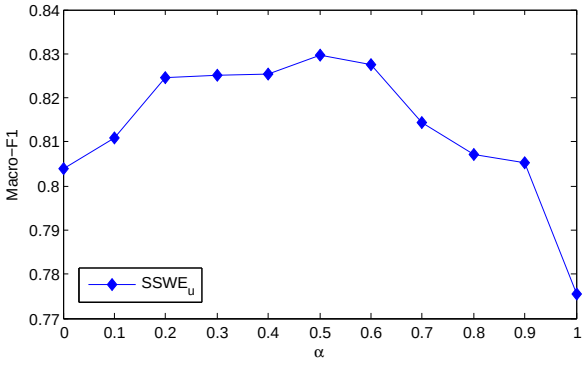


Figure 2: Macro-F1 of SSWE$_u$ on the development set of SemEval 2013 with different α.

Figure 2 shows the macro-F1 of SSWE$_u$ on positive/negative classification of tweets with different α on our development set. We can see that SSWE$_u$ performs better when α is in the range of [0.5, 0.6], which balances the syntactic context and sentiment information. The model with α=1 stands for C&W model, which only encodes the syntactic contexts of words. The sharp decline at α=1 reflects the importance of sentiment information in learning word embedding for Twitter sentiment classification.

Effect of Distant-supervised Data in SSWE$_u$ We investigate how the size of the distant-supervised data affects the performance of SSWE$_u$ feature for Twitter sentiment classification. We vary the number of distant-supervised tweets from 1 million to 12 million, increased by 1 million. We set the α of SSWE$_u$ as 0.5, according to the experiments shown in Figure 2. Results of positive/negative classification of tweets on our development set are given in Figure 3.

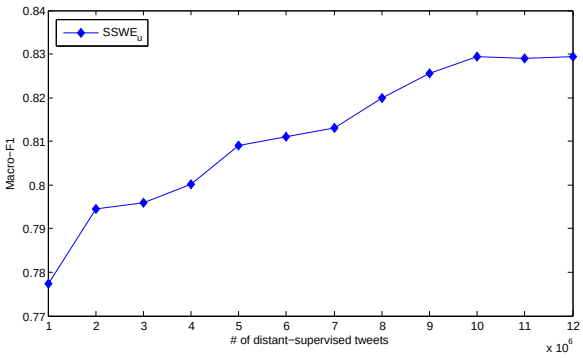


Figure 3: Macro-F1 of SSWE$_u$ with different size of distant-supervised data on our development set.

We can see that when more distant-supervised tweets are added, the accuracy of SSWE$_u$ consistently improves. The underlying reason is that when more tweets are incorporated, the word embedding is better estimated as the vocabulary size is larger and the context and sentiment information are richer. When we have 10 million distant-supervised tweets, the SSWE$_u$ feature increases the macro-F1 of positive/negative classification of tweets to 82.94% on our development set. When we have more than 10 million tweets, the performance remains stable as the contexts of words have been mostly covered.

4.2 Word Similarity of Sentiment Lexicons

The quality of SSWE has been implicitly evaluated when applied in Twitter sentiment classification in the previous subsection. We explicitly evaluate it in this section through word similarity in the em-

bedding space for sentiment lexicons. The evaluation metric is the accuracy of polarity consistency between each sentiment word and its top N closest words in the sentiment lexicon,

$$Accuracy = \frac{\sum_{i=1}^{\#Lex} \sum_{j=1}^{N} \beta(w_i, c_{ij})}{\#Lex \times N} \quad (10)$$

where $\#Lex$ is the number of words in the sentiment lexicon, w_i is the *i-th* word in the lexicon, c_{ij} is the *j-th* closest word to w_i in the lexicon with cosine similarity, $\beta(w_i, c_{ij})$ is an indicator function that is equal to 1 if w_i and c_{ij} have the same sentiment polarity and 0 for the opposite case. The higher accuracy refers to a better polarity consistency of words in the sentiment lexicon. We set N as 100 in our experiment.

Experiment Setup and Datasets We utilize the widely-used sentiment lexicons, namely *MPQA* (Wilson et al., 2005) and *HL* (Hu and Liu, 2004), to evaluate the quality of word embedding. For each lexicon, we remove the words that do not appear in the lookup table of word embedding. We only use unigram embedding in this section because these sentiment lexicons do not contain phrases. The distribution of the lexicons used in this paper is listed in Table 4.

Lexicon	Positive	Negative	Total
HL	1,331	2,647	3,978
MPQA	1,932	2,817	4,749
Joint	1,051	2,024	3,075

Table 4: Statistics of the sentiment lexicons. *Joint* stands for the words that occur in both *HL* and *MPQA* with the same sentiment polarity.

Results. Table 5 shows our results compared to other word embedding learning algorithms. The accuracy of *random* result is 50% as positive and negative words are randomly occurred in the nearest neighbors of each word. Sentiment-specific word embeddings (SSWE$_h$, SSWE$_r$, SSWE$_u$) outperform existing neural models (C&W, word2vec) by large margins. SSWE$_u$ performs best in three lexicons. SSWE$_h$ and SSWE$_r$ have comparable performances. Experimental results further demonstrate that sentiment-specific word embeddings are able to capture the sentiment information of texts and distinguish words with opposite sentiment polarity, which are not well solved in traditional neural models like C&W and word2vec. SSWE outperforms MVSA and ReEmb by exploiting more context information of words and sentiment information of sentences, respectively.

Embedding	HL	MPQA	Joint
Random	50.00	50.00	50.00
C&W	63.10	58.13	62.58
Word2vec	66.22	60.72	65.59
ReEmb(C&W)	64.81	59.76	64.09
ReEmb(w2v)	67.16	61.81	66.39
WVSA	68.14	64.07	67.12
SSWE$_h$	74.17	68.36	74.03
SSWE$_r$	73.65	68.02	73.14
SSWE$_u$	**77.30**	**71.74**	**77.33**

Table 5: Accuracy of the polarity consistency of words in different sentiment lexicons.

5 Conclusion

In this paper, we propose learning continuous word representations as features for Twitter sentiment classification under a supervised learning framework. We show that the word embedding learned by traditional neural networks are not effective enough for Twitter sentiment classification. These methods typically only model the context information of words so that they cannot distinguish words with similar context but opposite sentiment polarity (e.g. *good* and *bad*). We learn sentiment-specific word embedding (SSWE) by integrating the sentiment information into the loss functions of three neural networks. We train SSWE with massive distant-supervised tweets selected by positive and negative emoticons. The effectiveness of SSWE has been implicitly evaluated by using it as features in sentiment classification on the benchmark dataset in SemEval 2013, and explicitly verified by measuring word similarity in the embedding space for sentiment lexicons. Our unified model combining syntactic context of words and sentiment information of sentences yields the best performance in both experiments.

Acknowledgments

We thank Yajuan Duan, Shujie Liu, Zhenghua Li, Li Dong, Hong Sun and Lanjun Zhou for their great help. This research was partly supported by National Natural Science Foundation of China (No.61133012, No.61273321, No.61300113).

References

Luciano Barbosa and Junlan Feng. 2010. Robust sentiment detection on twitter from biased and noisy data. In *Proceedings of International Conference on Computational Linguistics*, pages 36–44.

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

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Analysis and Machine Intelligence*.

Yoshua Bengio. 2013. Deep learning of representations: Looking forward. *arXiv preprint arXiv:1305.0445*.

Dmitriy Bespalov, Bing Bai, Yanjun Qi, and Ali Shokoufandeh. 2011. Sentiment classification based on supervised latent n-gram analysis. In *Proceedings of the Conference on Information and Knowledge Management*, pages 375–382.

Dmitriy Bespalov, Yanjun Qi, Bing Bai, and Ali Shokoufandeh. 2012. Sentiment classification with supervised sequence embedding. In *Machine Learning and Knowledge Discovery in Databases*, pages 159–174. Springer.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Annual Meeting of the Association for Computational Linguistics*, volume 7.

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

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Enhanced sentiment learning using twitter hashtags and smileys. In *Proceedings of International Conference on Computational Linguistics*, pages 241–249.

Xiaowen Ding, Bing Liu, and Philip S Yu. 2008. A holistic lexicon-based approach to opinion mining. In *Proceedings of the International Conference on Web Search and Data Mining*, pages 231–240.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, pages 2121–2159.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

Ronen Feldman. 2013. Techniques and applications for sentiment analysis. *Communications of the ACM*, 56(4):82–89.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 42–47.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. *Proceedings of International Conference on Machine Learning*.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, pages 1–12.

Karl Moritz Hermann and Phil Blunsom. 2013. The role of syntax in vector space models of compositional semantics. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 894–904.

Ming Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 168–177.

Xia Hu, Jiliang Tang, Huiji Gao, and Huan Liu. 2013. Unsupervised sentiment analysis with emotional signals. In *Proceedings of the International World Wide Web Conference*, pages 607–618.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. *The Proceeding of Annual Meeting of the Association for Computational Linguistics*, 1:151–160.

Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. 2011. Twitter sentiment analysis: The good the bad and the omg! In *The International AAAI Conference on Weblogs and Social Media*.

Igor Labutov and Hod Lipson. 2013. Re-embedding words. In *Annual Meeting of the Association for Computational Linguistics*.

Kun-Lin Liu, Wu-Jun Li, and Minyi Guo. 2012. Emoticon smoothed language models for twitter sentiment analysis. In *The Association for the Advancement of Artificial Intelligence*.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *The Conference on Neural Information Processing Systems*.

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

Andriy Mnih and Geoffrey E Hinton. 2009. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088.

Saif M Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. *Proceedings of the International Workshop on Semantic Evaluation*.

Preslav Nakov, Sara Rosenthal, Zornitsa Kozareva, Veselin Stoyanov, Alan Ritter, and Theresa Wilson. 2013. Semeval-2013 task 2: Sentiment analysis in twitter. In *Proceedings of the International Workshop on Semantic Evaluation*, volume 13.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of Language Resources and Evaluation Conference*, volume 2010.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 79–86.

Richard Socher, Eric H Huang, Jeffrey Pennington, Andrew Y Ng, and Christopher D Manning. 2011a. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. *The Conference on Neural Information Processing Systems*, 24:801–809.

Richard Socher, Cliff C Lin, Andrew Ng, and Chris Manning. 2011b. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the International Conference on Machine Learning*, pages 129–136.

Richard Socher, J. Pennington, E.H. Huang, A.Y. Ng, and C.D. Manning. 2011c. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Conference on Empirical Methods in Natural Language Processing*, pages 151–161.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic Compositionality Through Recursive Matrix-Vector Spaces. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013a. Parsing with compositional vector grammars. In *Annual Meeting of the Association for Computational Linguistics*.

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

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307.

Mike Thelwall, Kevan Buckley, and Georgios Paltoglou. 2012. Sentiment strength detection for the social web. *Journal of the American Society for Information Science and Technology*, 63(1):163–173.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. *Annual Meeting of the Association for Computational Linguistics*.

Peter D Turney. 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 417–424.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 90–94.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 347–354.

Ainur Yessenalina and Claire Cardie. 2011. Compositional matrix-space models for sentiment analysis. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, pages 172–182.

Jichang Zhao, Li Dong, Junjie Wu, and Ke Xu. 2012. Moodlens: an emoticon-based sentiment analysis system for chinese tweets. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for chinese word segmentation and pos tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 647–657.

Towards a General Rule for Identifying Deceptive Opinion Spam

Jiwei Li[1], **Myle Ott**[2], **Claire Cardie**[2], **Eduard Hovy**[1]
[1]Language Technology Institute, Carnegie Mellon University, Pittsburgh, P.A. 15213, USA
[2]Department of Computer Science, Cornell University, Ithaca, N.Y., 14853, USA
bdlijiwei@gmail.com, myleott@cs.cornell.edu
cardie@cs.cornell.edu, ehovy@andrew.cmu.edu

Abstract

Consumers' purchase decisions are increasingly influenced by user-generated online reviews. Accordingly, there has been growing concern about the potential for posting deceptive opinion spam—fictitious reviews that have been deliberately written to sound authentic, to deceive the reader. In this paper, we explore generalized approaches for identifying online deceptive opinion spam based on a new gold standard dataset, which is comprised of data from three different domains (i.e. Hotel, Restaurant, Doctor), each of which contains three types of reviews, i.e. *customer generated truthful reviews*, *Turker generated deceptive reviews* and *employee (domain-expert) generated deceptive reviews*. Our approach tries to capture the general difference of language usage between deceptive and truthful reviews, which we hope will help customers when making purchase decisions and review portal operators, such as TripAdvisor or Yelp, investigate possible fraudulent activity on their sites.[1]

1 Introduction

Consumers increasingly rely on user-generated online reviews when making purchase decision (Cone, 2011; Ipsos, 2012). Unfortunately, the ease of posting content to the Web, potentially anonymously, creates opportunities and incentives for unscrupulous businesses to post *deceptive opinion spam*—fictitious reviews that are deliberately written to sound authentic, in order to deceive the reader.[2] Accordingly, there appears to be widespread and growing concern among both businesses and the public about this potential abuse (Meyer, 2009; Miller, 2009; Streitfeld, 2012; Topping, 2010; Ott, 2013).

Existing approaches for spam detection are usually focused on developing supervised learning-based algorithms to help users identify deceptive opinion spam, which are highly dependent upon high-quality gold-standard labeled data (Jindal and Liu, 2008; Jindal et al., 2010; Lim et al., 2010; Wang et al., 2011; Wu et al., 2010). Studies in the literature rely on a couple of approaches for obtaining labeled data, which usually fall into two categories. The first relies on the judgements of human annotators (Jindal et al., 2010; Mukherjee et al., 2012). However, recent studies show that deceptive opinion spam is not easily identified by human readers (Ott et al., 2011). An alternative approach, as introduced by Ott et al. (2011), crowdsourced deceptive reviews using Amazon Mechanical Turk.[3] A couple of follow-up works have been introduced based on Ott et al.'s dataset, including estimating prevalence of deception in online reviews (Ott et al., 2012), identification of negative deceptive opinion spam (Ott et al., 2013), and identifying manipulated offerings (Li et al., 2013b).

Despite the advantages of soliciting deceptive gold-standard material from Turkers (it is easy, large-scale, and affordable), it is unclear whether Turkers are representative of the general population that generate fake reviews, or in other words, Ott et al.'s data set may correspond to only one type of online deceptive opinion spam — fake reviews generated by people who have never been to offerings or experienced the entities. Specifically, according to their findings (Ott et al., 2011;

[1]Dataset available by request from the first author.

[2]Manipulating online reviews may also have legal consequences. For example, the Federal Trade Commission (FTC) has updated their guidelines on the use of endorsements and testimonials in advertising to suggest that posting deceptive reviews may be unlawful in the United States (FTC, 2009).

[3]http://www.mturk.com

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1566–1576,
Baltimore, Maryland, USA, June 23-25 2014.

Li et al., 2013a), truthful hotel reviews encode more spatial details, characterized by terms such as "bathroom" and "location", while deceptive reviews talk about general concepts such as why or with whom they went to the hotel. However, a hotel can instead solicit fake reviews from their employees or customers who possess substantial domain knowledge to write fake reviews and encode more spatial details in their lies. Indeed, cases have been reported where hotel owners bribe guests in return for good reviews on TripAdvisor[4], or companies ordered employees to pretend they were satisfied customers and write glowing reviews of its face-lift procedure on Web sites.[5] The domain knowledge possessed by domain experts enables them to craft reviews that are much more difficult for classifiers to detect, compared to the crowdsourced fake reviews.

Additionally, existing supervised algorithms in the literature are usually narrowed to one specific domain and heavily rely on domain-specific vocabulary. For example, classifiers assign high weights to domain-specific terms such as "hotel", "rooms", or even the name of the hotels such as "Hilton" when trained on reviews on hotels. It is unclear whether these classifiers will perform well at detecting deception in other domains, e.g., Restaurant or Doctor reviews. Even in a single domain, e.g., Hotel, classifiers trained from reviews of one city (e.g., Chicago) may not be effective if directly applied to reviews from other cities (e.g., New York City) (Li et al., 2013b). In the examples in Table 1, we trained a linear SVM classifier on Ott's Chicago-hotel dataset on *unigram* features and tested it on a couple of different domains (the details of data acquisition are illustrated in Section 3). Good performance is obtained on Chicago-hotel reviews (Ott et al., 2011), but not as good on New York City ones. The performance is reasonable in Restaurant reviews due to the many shared properties among restaurants and hotels, but suffers in Doctor settings.

In this paper, we try to obtain a deeper understanding of the general nature of deceptive opinion spam. One contribution of the work presented here is the creation of the cross-domain (i.e., Hotel, Restaurant and Doctor) gold-standard dataset.

	Accuracy	Precision	Recall	F1
NYC-Hotel	0.799	0.794	0.758	0.766
Chicago-Restaurant	0.785	0.813	0.742	0.778
Doctor	0.550	0.537	0.725	0.617

Table 1: SVM performance on datasets for a classifier trained on Chicago hotel review based on Unigram feature.

In contrast to existing work (Ott et al., 2011; Li et al., 2013b), our new gold standard includes three types of reviews: *domain expert deceptive opinion spam* (*Employee*), *crowdsourced deceptive opinion spam* (*Turker*), and *truthful Customer reviews* (*Customer*). In addition, some of domains contain both positive (**P**) and negative (**N**) reviews.[6]

To explore the general rule of deceptive opinion spam, we extended SAGE Model (Eisenstein et al., 2011), a bayesian generative approach that can capture the multiple generative facets (i.e., deceptive vs truthful, positive vs negative, experienced vs non-experienced, hotel vs restaurant vs doctor) in the text collection. We find that more general features, such as LIWC and POS, are more robust when modeled using SAGE, compared with just bag-of-words.

We additionally make theoretical contributions that may shed light on a longstanding debate in the literature about deception. For example, in contrast to existing findings that highlight the lack of spatial detail in deceptive reviews (Ott et al., 2011; Li et al., 2013b), we find that a lack of spatial detail may not be a universal cue to deception, since it does not apply to fake reviews written by domain experts. Instead, our finding suggest that other linguistic features may offer more robust cues to deceptive opinion spam, such as overly highlighted sentiment in the review or the overuse of first-person singular pronouns.

The rest of this paper is organized as follows. In Section 2, we briefly go over related work. We describe the creation of our data set in Section 3 and present our model in Section 4. Experimental results are shown in Section 5. We present analysis of general cues to deception in Section 6 and conclude this paper in Section 7.

[4]http://www.dailymail.co.uk/travel/article-2013391/Tripadvisor-Hotel-owners-bribe-guests-return-good-reviews.html

[5]http://www.nytimes.com/2009/07/15/technology/internet/15lift.html?_r=0

[6]For example, a hotel manager could hire people to write positive reviews to increase the reputation of his own hotel or post negative ones to degrade his competitors. Identifying positive/negative opinion spam is explored in (Ott et al., 2011; Ott et al., 2013)

2 Related Work

Spam has been historically studied in the contexts of Web text (Gyöngyi et al., 2004; Ntoulas et al., 2006) or email (Drucker et al., 1999). Recently there has been increasing concern about deceptive opinion spam (Jindal and Liu, 2008; Ott et al., 2011; Wu et al., 2010; Mukherjee et al., 2013b; Wang et al., 2012).

Jindal and Liu (2008) first studied the deceptive opinion problem and trained models using features based on the review text, reviewer, and product to identify duplicate opinions, i.e., opinions that appear more than once in the corpus with similar contexts. Wu et al. (2010) propose an alternative strategy to detect deceptive opinion spam in the absence of a gold standard. Yoo and Gretzel (2009) gathered 40 truthful and 42 deceptive hotel reviews and manually compare the linguistic differences between them. Ott et al. created a gold-standard collection by employing Turkers to write fake reviews, and follow-up research was based on their data (Ott et al., 2012; Ott et al., 2013; Li et al., 2013b; Feng and Hirst, 2013). For example, Song et al. (2012) looked into syntactic features from Context Free Grammar parse trees to improve the classifier performance. A step further, Feng and Hirst (2013) make use of degree of *compatibility* between the personal experiment and a collection of reference reviews about the same product rather than simple textual features.

In addition to exploring text or linguistic features in deception, some existing work looks into customers' behavior to identify deception (Mukherjee et al., 2013a). For example, Mukherjee et al. (2011; 2012) delved into group behavior to identify group of reviewers who work collaboratively to write fake reviews. Qian and Liu (2013) identified multiple user IDs that are generated by the same author, as these authors are more likely to generate deceptive reviews.

In the psychological literature, researchers have looked into possible linguistic cues to deception (Newman et al., 2003), such as *decreased spatial detail*, which is consistent with theories of reality monitoring (Johnson and Raye, 1981), increased negative emotion terms (Newman et al., 2003), or the writing style difference between informative (truthful) and imaginative (deceptive) writings in (Rayson et al., 2001). The former typically consists of more nouns, adjectives, prepositions, determiners, and coordinating conjunctions, while the latter consists of more verbs, adverbs, pronouns, and pre-determiners.

SAGE (Sparse Additive Generative Model): SAGE is an generative bayesian approach introduced by Eisenstein et al. (2011), which can be viewed as an combination of topic models (Blei et al., 2003) and generalized additive models (Hastie and Tibshirani, 1990). Unlike other derivatives of topic models, SAGE drops the Dirichlet-multinomial assumption and adopts a Laplacian prior, triggering sparsity in topic-word distribution. The reason why SAGE is tailored for our task is that SAGE constructs multi-faceted latent variable models by simply adding together the component vectors rather than incorporating multiple switching latent variables in multiple facets.

3 Dataset Construction

In this section, we report our efforts to gather gold-standard opinion spam datasets. Our datasets contain the following domains, namely Hotel, Restaurant, and Doctor.

3.1 *Turker* set, using Mechanical Turk

Crowdsourcing services such as AMT greatly facilitate large-scale data annotation and collection efforts. Anyone with basic programming skills can create Human Intelligence Tasks (HITs) and access a marketplace of anonymous online workers (Turkers) willing to complete the tasks. We borrowed some rules used by Ott et al. to create their dataset, such as restricting task to Turkers located in the United States, and who maintain an approval rating of at least 90%.

Hotel-Turker : We directly borrowed datasets from Ott[7] and Li.[8]

Restaurant-Turker : We gathered 20 positive (P) deceptive reviews for each of 10 of the most popular restaurants in Chicago, for a total of 200 positive deceptive restaurant reviews.

Doctor-Turker : We gathered a total number of 200 positive reviews from Turkers.

3.2 *Employee* set, by domain experts

We seek deceptive opinion spam written by people with expert-level domain knowledge. It is not appropriate to use crowdsourcing to obtain this data,

[7] http://myleott.com/op_spam/

[8] http://www.cs.cmu.edu/~jiweil/html/four_city.html

	Turker	*Expert*	*Customer*
Hotel (P/N)	400/400	140/140	400/400
Restaurant (P/N)	200/0	120/0	200/200
Doctor (P/N)	200/0	32/0	200/0

Table 2: Statistics for our dataset.

so instead we solicit reviews written by employees in each domain.

Hotel-Employee: We asked two hotel employees from each of seven hotels (14 employees total) each to write 10 deceptive positive-sentiment reviews of their own hotel, and 10 deceptive negative-sentiment reviews of their biggest local competitor's hotel. In total, we obtained 280 deceptive reviews of 14 hotels, including a balanced mix of positive- and negative-sentiment reviews.

Restaurant-Employee: We asked employees from selected restaurants (a waiter/waitress or cook) to each write positive-sentiment reviews of their restaurant.

Doctor-Employee: We asked real doctors to write positive fake reviews about themselves. In total we obtained 32 reviews from 15 doctors.

3.3 *Customer* set from Actual Customers

Hotel-Customer: We borrowed from Ott et al.'s dataset.

Restaurant/Doctor-Customer: We solicited data by matching a set of truthful reviews as Ott et al. did in collecting truthful hotel reviews.

3.4 Summary for Data Creation

Statistics for our data set is presented in Table 2. Due to the difficulty in obtaining gold-standard data in the literature, there is no doubt that our data set is not perfect. Some parts are missing, some are unbalanced, participants in the survey may not be representative of the general population. However, as far as we know, this is the most comprehensive dataset for deceptive opinion spam so far, and may to some extent shed insights on the nature of online deception.

4 Feature-based Additive Model

In this section, we briefly describe our model. Since mathematics are not the main theme of this paper, we omit the exact details for inference, which can be found in (Eisenstein et al., 2011).

Before describing the model in detail, we note the following advantages of the SAGE model, and our reasons for using it in this paper:

1. the "additive" nature of SAGE allows a better understanding of which features contribute most to each type of deceptive review and how much each such feature contributes to the final decision jointly. If we instead use SVM, for example, we would have to train classifiers one by one (due to the distinct features from different sources) to draw conclusions regarding the differences between Turker vs Expert vs truthful reviews, positive expert vs negative expert reviews, or reviews from different domains. This would not only become intractable, but would make the conclusions less clear.

2. For cross-domain classification task, standard machine learning approaches may suffer due to domain-specific properties (See Section 5.2).

4.1 Model

In SAGE, each term w is drawn from a distribution proportional to $\exp(m^{(w)} + \eta_{y_d}^{(T)(w)} + \eta_{z_n}^{(A)(w)} + \eta_{y_d,z_n}^{(I)(w)})$, where $m^{(w)}$ is the observed background term frequency, η_{y_d}, η_{z_n} and η_{y_d,z_n} denote the log frequency deviation representing topic z_n, facet y_d, and the second-order interaction part respectively. Superscripts T, A and I respectively denote the index of the topic, facet, and second-order interaction. In our task, we adapt the SAGE model as follows:

$$\begin{aligned} Y = \{&y_{Sentiment} \in \{\text{positive, negative}\}, \\ &y_{Domain} \in \{\text{hotel, restaurant, doctor}\}, \\ &y_{Source} \in \{\text{employee, turker, customer}\}\} \end{aligned}$$

We model three η's, one for each type of y. Let i, j, k denote the index of the different types of y, so that each term w is drawn as follows:

$$\begin{aligned} P(w|i,j,k) \propto \exp(&m^{(w)} + \eta_{y_{Sentiment}}^{(i)(w)} \\ &+\eta_{y_{Domain}}^{(j)(w)} + \eta_{y_{Scource}}^{(k)(w)} + \textit{higher order}) \end{aligned}$$

where the *higher order* parts denote the interactions between different facets.

In our approach each document-level feature f is drawn from the following distribution:

$$\begin{aligned} P(f|i,j,k) \propto \exp(&m^{(f)} + \eta_{y_{Sentiment}}^{(i)(f)} + \eta_{y_{Domain}}^{(j)(f)} \\ &+ \eta_{y_{Scource}}^{(k)(f)} + higher\ order) \end{aligned} \quad (1)$$

where $m^{(f)}$ can be interpreted as the background value of feature f. For each review d, the probability that it is drawn from facets with index i, j, k is as follows:

$$P(d|i,j,k) = \prod_{f \in d} P(f|i,j,k) \prod_{w \in d} P(w|i,j,k) \quad (2)$$

In the training process, parameters $\eta_y^{(w)}$ and $\eta_y^{(f)}$ are to be learned by maximizing the posterior distribution following the original SAGE training procedure. For prediction, we estimate y_{Source} for each document given all or part of $\eta_y^{(w)}$ and $\eta_y^{(f)}$ as follows:

$$y_{Source} = \underset{y'_{Source}}{\operatorname{argmax}} P(d|y'_{Source}, y_{Sentiment}, y_{Domain}),$$

where we assume $y_{Sentiment}$ and y_{Domain} are given for each document d. Note that we assume conditional independence between features and words given y, similar to other topic models (Blei et al., 2003). Notably, our revised SAGE model degenerates into a model similar to Generalized Additive Model (Hastie and Tibshirani, 1990) when word features are not considered.

5 Experiments

In this section, we report our experimental results. We first restrict experiments to the within-domain task and see what features most characterize the deceptive reviews, and how. We later extend it to cross domains to explore a more general classifier of deceptive opinion spam.

5.1 Intra-Domain Classification

We explore the effect of both domain experts and crowdsourcing workers on intra-domain deception. Specifically, we reframe it as a **intra-domain multi-class classification task**, where given the labeled training data from one domain, we learn a classifier to classify reviews according to their source, i.e., *Employee*, *Turker* and *Customer*. Since the machine learning classifier is trained and tested within the same domain, $\eta_{y_{Domain}}^{(j)(w)}$ and $\eta_{y_{Domain}}^{(i)(f)}$ are not considered here.

We use a *One-Versus-Rest* (OvR) scheme, in which we train m classifiers using SAGE, such that each classifier f_i, for $i \in [1, m]$, is trained to distinguish between class i on the one hand, and all classes except i on the other. To make an m-way decision, we then choose the class c with the most confident prediction. OvR approaches have been shown to produce state-of-art performance compared to other multi-class approaches such as *Multinomial Naive Bayes* or *One-Versus-One* classification scheme. We train the OvR classifier on three sets of features, *LIWC*, *Unigram*, and *POS*.[9]

Multi-class classification results are given at Table 3. We report both OvR performance and the performance of three One-versus-One binary classifiers, trained to distinguish between each pair of classes. In particular, the three-class classifier is around 69% accurate at distinguishing between *Employee*, *Customer*, and *Turker* for each of the domains using Unigram, significantly higher than random guess. We also observe that each of the three *One-versus-One* binary classifications performs significantly better than chance, suggesting that *Employee*, *Customer*, and *Turker* are in fact three different classes. In particular, the two-class classifier is around 0.76 accurate in distinguishing between *Turker* and *Employee* reviews, despite both kinds of reviews being deceptive opinion spam.

Best performance is achieved on Unigram features, constantly outperforming LIWC and POS features in both three-class and two-class settings in the hotel domain. Similar results are observed for restaurant and doctor domains and details are excluded for brevity. This suggests that a universal set of keyword-based deception cues (e.g., LIWC) is not the best approach for Intra-Domain Classification. Similar results were also reported in previous work (Ott et al., 2012; Ott, 2013).

5.2 Cross-domain Classification

In this subsection, we frame our problem as a *domain adaptation task* (Pan and Yang, 2010). Again, we explore 3 feature sets: *LIWC*, *Unigram* and *POS*. We train a classifier on hotel reviews, and evaluate the performance on other domains. For simplicity, we focus on **truthful** (*Customer*) versus **deceptive** (*Turker*) binary classification rather than a multi-class classification.

We report results from SAGE and SVM[10] in Table 4. We first observe that classifiers trained on hotel reviews apply well in the restaurant domain, which is reasonable due to the many shared prop-

[9] Part-of-speech tags were assigned based on Stanford Parser `http://nlp.stanford.edu/software/lex-parser.shtml`

[10] We use SVMlight (Joachims, 1999) to train our linear SVM classifiers

Domain	Setting	Features		*Customer*		*Employee*		*Turker*	
			A	P	R	P	R	P	R
Hotel	Three-Class	Unigram	0.664	0.678	0.669	0.589	0.610	0.641	0.582
		LIWC	0.602	0.617	0.613	0.541	0.598	0.590	0.511
		POS	0.517	0.532	0.669	0.481	0.479	0.482	0.416
	Customer vs *Turker*	Unigram	0.818	0.812	0.840	-	-	0.820	0.809
		LIWC	0.764	0.774	0.771	-	-	0.723	0.749
		POS	0.729	0.748	0.692	-	-	0.707	0.759
	Customer vs *Employee*	Unigram	0.799	0.832	0.784	0.804	0.820	-	-
		LIWC	0.732	0.746	0.751	0.714	0.722	-	-
		POS	0.728	0.713	0.742	0.707	0.754	-	-
	Employee vs *Turker*	Unigram	0.762	-	-	0.786	0.806	0.826	0.794
		LIWC	0.720	-	-	0.728	0.726	0.698	0.739
		POS	0.701	-	-	0.688	0.710	0.701	0.697
Restaurant	Three-Class	Unigram	0.647	0.692	0.725	0.625	0.648	0.686	0.702
	Customer vs *Turker*		0.817	0.842	0.816	-	-	0.804	0.812
	Customer vs *Employee*		0.785	0.790	0.814	0.769	0.826	-	-
	Employee vs *Turker*		0.774	-	-	0.784	0.804	0.802	0.763
Doctor	*Customer* vs *Turker*		0.745	0.772	0.701	-	-	0.752	0.718

Table 3: Within-domain multi-class classifier performance.

Model	Features	Domain	A	P	R	F1	Domain	A	P	R	F1
SVM	Unigram	Restaurant	0.785	0.813	0.742	0.778	Doctor	0.550	0.537	0.725	0.617
	LIWC	Restaurant	0.745	0.692	0.840	0.759	Doctor	0.521	0.512	0.965	0.669
	POS	Restaurant	0.735	0.697	0.815	0.751	Doctor	0.540	0.521	0.975	0.679
SAGE	Unigram	Restaurant	0.770	0.793	0.750	0.784	Doctor	0.520	0.547	0.705	0.616
	LIWC	Restaurant	0.742	0.728	0.749	0.738	Doctor	0.647	0.650	0.608	0.628
	POS	Restaurant	0.746	0.732	0.687	0.701	Doctor	0.634	0.623	0.682	0.651

Table 4: Classifier performance in cross-domain adaptation.

erties among restaurants and hotels. Among three types of features, Unigram still performs best. POS and LIWC features are also robust across domains.

In the doctor domain, we observe that models trained on Unigram features from the hotels domain do not generalize well to doctor reviews, and the performance is a little bit better than random guess with only 0.55 accuracy. For SVM, models trained on POS and LIWC features achieve even lower accuracy than Unigram. POS and LIWC features obtain around 0.5 precision and 1.0 recall, indicating that all doctor reviews are classified as deceptive by the classifier. One plausible explanation could be doctor reviews generally encode some type of positive-weighted (deceptive) features more than hotel reviews and these types of features dominate the decision making procedures, leading all reviews to be classified as deceptive.

Tables 5 and 6 give the top weighted LIWC and POS features. We observe that many features are indeed shared among doctor and hotel domains. Notably, POS features are more robust than LIWC as more shared features are observed. As domain specific properties will be considered in the interaction part (η^{LIWC}_{domain} and η^{POS}_{domain}) of the additive model, SAGE achieve much better results than SVM, and is around 0.65 accurate in the cross-domain task.

LIWC (hotel)		LIWC (doctor)	
deceptive	truthful	deceptive	truthful
i	AllPct	Sixletters	present
family	number	past	AllPct
pronoun	hear	work	social
Sixletters	we	health	shehe
see	space	i	number
posemo	dash	friend	time
certain	human	posemo	we
leisure	exclusive	feel	you
future	past	perceptual	negemo
perceptual	home	leisure	Period
feel	otherpunct	insight	relativ
comma	negemo	comma	ingest
cause	dash	future	money

Table 5: Top weighted LIWC features for Turker vs Customer in Doctor and Hotel reviews. Blue denotes shared positive (deceptive) features and red denotes negative (truthful) features.

6 General Linguistic Cues of Deceptive Opinion Spam

In this section, we examine a number of general POS and LIWC features that may shed light on a general rule for identifying deceptive opinion

Figure 1: Visualization of the η for POS features: Horizontal axes correspond to the values η and are NORMALIZED from the log-frequency function.

POS (hotel)		POS (doctor)	
deceptive	truthful	deceptive	truthful
PRP$	CD	VBD	CD
PRP	RRB	NNP	VBZ
VB	LRB	VB	VBP
TO	CC	TO	FW
NNP	NNS	VBG	RRB
VBG	RP	PRP$	LRB
MD	VBN	JJS	RB
VBP	IN	JJ	LS
RB	EX	WRB	PDT
JJS	VBZ	PRP	VBN

Table 6: Top weighted POS features for Turker vs Customer in Doctor and Hotel reviews. Blue denotes shared positive (deceptive) features and red denotes negative (truthful) features.

spam. Our modified SAGE model provides us with a tailored tool for this analysis. Specifically, each feature f is associated with a background value m^f. For each facet A, η_A^f, presents the facet-specific preference value for feature f. Note that sentiments are separated into positive and negative dimensions, which is necessary because hotel employee authors wrote positive-sentiment reviews when reviewing their own hotels, and negative-sentiment reviews when reviewing their competitors' hotels.

6.1 POS features

Early findings in the literature (Rayson et al., 2001; Buller and Burgoon, 1996; Biber et al., 1999) found that informative (truthful) writings typically consist of more nouns, adjectives, prepositions, determiners, and coordinating conjunctions, while imaginative (deceptive) writing consist of more verbs, adverbs, pronouns, and pre-determiners (with a few exceptions). Our findings with POS features are largely in agreement with these findings when distinguishing between Turker and Customer reviews, but are violated in the Employee set.

We present the eight types of POS features in Figure 1, namely, N (Noun), JJ (Adjective), IN (Preposition or subordinating conjunction) and DT (Determiner), V (Verb), RB (Adverb), PRP (Pronouns, both personal and possessive) and PDT (Pre-Determiner).

From Figures 1(a)(b)(e)(f), we observe that with the exception of PDT, the word frequency of which is too small to draw a conclusion, *Turker* and *Customer* reviews exhibit linguistic patterns in agreement with previous findings in the literature, where truthful reviews (*Customer*) tend to include more N, JJ, IN and DT, while deceptive writings tend to encode more V, RB and PRP.

However, in the case of the *Employee-Positive* dataset, which is equally deceptive, most of these rules are violated. Notably, reviews from the *Employee-Positive* set did not encode fewer N, JJ and DT terms, as expected (see Figures 1(a)(c)). Instead, they encode even more N, JJ and DT vocabularies than truthful reviews from the *Customer* reviews. Also, fewer V and RB are found in *Employee-Positive* reviews compared with *Customer* reviews (see Figures 1(e)(g)).

One explanation for these observations is that informative (truthful) writing tends to be more introductory and descriptive, encoding more concrete details, when compared with imaginary writings. As domain experts possess considerable knowledge of their own offerings, they highlight

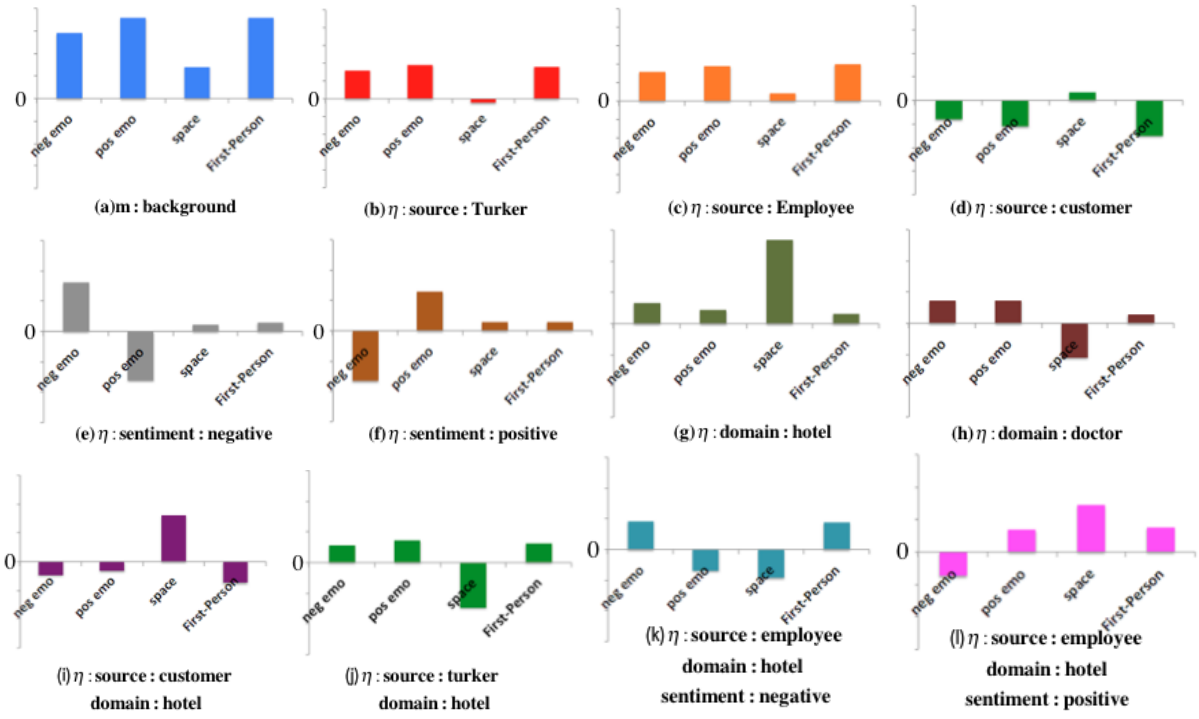


Figure 2: Visualization of the η for LIWC features: Horizontal axes correspond to the values η and are normalized from the log-frequency function.

the details and their lies may be even more informative and descriptive than those generated by real customers! This explains why *Employee-Positive* contains more N, IN and DT. Meanwhile, as domain experts are engaged more in talking about the details, they inevitably overlook other information, possibly leading to fewer V and RB.

For *Employee-Positive* reviews, shown in Figures 1(d)(h), it turns out that domain experts do not compensate for their lack of prior experience when writing negative reviews for competitors' offerings, as we will see again with LIWC features in the next subsection.

6.2 LIWC features

We explore 3 LIWC categories (from left to right in subfigures of Figure 2): sentiment (*neg emo* and *pos emo*), spatial detail (*space*), and first-person singular pronouns (*first-person*).

Space: Note that spatial details are more specific in the *Hotel* and *Restaurant* domains, which is reflected in the high positive value of $\eta_{domain}^{Hotel,space}$ (see Figure 2(g)) and negative value of $\eta_{domain}^{Doctor,space}$ (see Figure 2(h)). It illustrates how domain-specific details can be predictive of deceptive text. Similarly predictive LIWC features are *home* for the *Hotel* domain, *ingest* for the *Restaurant* domain, and *health* and *body* for the *Doctor* domain.

In Figure 2(i)(j)(k)(l), we can easily see that both actual customers and domain experts encode more spatial details in their reviews (positive value of η), which is in agreement with our expectation. This further demonstrates that a lack of spatial details would not be a general cue for deception. Moreover, it appears that general domain expertise does not compensate for the lack of prior experience when writing deceptive negative reviews for competitors' hotels, as demonstrated by the lack of spatial details in the negative-sentiment reviews by employees shown in Figure 2(k).

Sentiment: According to our findings, the presence of sentiment is a general cue to deceptive opinion spam, as observed when comparing Figure 2(b) to Figure 2(c) and (d). Participants, both Employees and Turkers, tend to exaggerate sentiment, and include more sentiment-related vocabularies in their lies. In other words, positive deceptive reviews were generally more positive and negative deceptive reviews were more negative in sentiment when compared with the truthful reviews generated by actual customers. A similar pattern can also be observed when comparing Figure 2(i) to Figure 2(j).

First-Person Singular Pronouns: The literature also associates deception with decreased usage of first-person singular pronouns, an effect attributed to psychological distancing, whereby deceivers talk less about themselves due either to a lack of personal experience, or to detach themselves from the lie (Newman et al., 2003; Zhou et al., 2004; Buller et al., 1996; Knapp and Comaden, 1979). However, according to our findings, we find the opposite to hold. Increased first person singular is an apparent indicator of deception, when comparing Figure 2(b) to 2(c) and 2(e). We suspect that this relates to an effect observed in previous studies of deception, where liars inadvertently undermine their lies by overemphasizing aspects of their deception that they believe reflect credibility (Bond and DePaulo, 2006; DePaulo et al., 2003). One interpretation for this phenomenon would be that deceivers try to overemphasize their physical presence because they believe that this increases their credibility.

7 Conclusion and Discussion

In this work, we have developed a multi-domain large-scale dataset containing gold-standard deceptive opinion spam. It includes reviews of Hotels, Restaurants and Doctors, generated through crowdsourcing and domain experts. We study this data using SAGE, which enables us to make observations about the respects in which truthful and deceptive text differs. Our model includes several domain-independent features that shed light on these differences, which further allows us to formulate some general rules for recognizing deceptive opinion spam.

We also acknowledge several important caveats to this work. By soliciting fake reviews from participants, including crowd workers and domain experts, we have found that is possible to detect fake reviews with above-chance accuracy, and have used our models to explore several psychological theories of deception. However, it is still very difficult to estimate the practical impact of such methods, as it is very challenging to obtain gold-standard data in the real world. Moreover, by soliciting deceptive opinion spam in an artificial environment, we are endorsing the deception, which may influence the cues that we observe (Feeley and others, 1998; Frank and Ekman, 1997; Newman et al., 2003; Ott, 2013). Finally, it may be possible to train people to tell more convincing lies. Many of the characteristics regarding fake review generation might be overcome by well-trained fake review writers, which would results in opinion spam that is harder for detect. Future work may wish to consider some of these additional challenges.

8 Acknowledgement

We thank Wenjie Li and Xun Wang for useful discussions and suggestions. This work was supported in part by National Science Foundation Grant BCS-0904822, a DARPA Deft grant, as well as a gift from Google. We also thank the ACL reviewers for their helpful comments and advice.

References

Douglas Biber, Stig Johansson, Geoffrey Leech, Susan Conrad, Edward Finegan, and Randolph Quirk. 1999. *Longman grammar of spoken and written English*, volume 2. MIT Press.

David Blei, Andrew Ng, and Michael Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Charles Bond and Bella DePaulo. 2006. Accuracy of deception judgments. *Personality and Social Psychology Review*, 10(3):214–234.

David B Buller and Judee K Burgoon. 1996. Interpersonal deception theory. *Communication theory*, 6(3):203–242.

David B Buller, Judee K Burgoon, Aileen Buslig, and James Roiger. 1996. Testing interpersonal deception theory: The language of interpersonal deception. *Communication theory*, 6(3):268–289.

Paul-Alexandru Chirita, Jörg Diederich, and Wolfgang Nejdl. 2005. Mailrank: using ranking for spam detection. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 373–380. ACM.

Cone. 2011. 2011 Online Influence Trend Tracker. http://www.coneinc.com/negative-reviews-online-reverse-purchase-decisions, August.

Bella DePaulo, James Lindsay, Brian Malone, Laura Muhlenbruck, Kelly Charlton, and Harris Cooper. 2003. Cues to deception. *Psychological bulletin*, 129(1):74.

Harris Drucker, Donghui Wu, and Vladimir Vapnik. 1999. Support vector machines for spam categorization. *Neural Networks, IEEE Transactions on*, 10(5):1048–1054.

Jacob Eisenstein, Amr Ahmed, and Eric P Xing. 2011. Sparse additive generative models of text. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1041–1048.

Thomas Feeley. 1998. The behavioral correlates of sanctioned and unsanctioned deceptive communication. *Journal of Nonverbal Behavior*, 22(3):189–204.

Vanessa Feng and Graeme Hirst. 2013. Detecting deceptive opinions with profile compatibility. In *Proceedings of the 6th International Joint Conference on Natural Language Processing, Nagoya, Japan*, pages 14–18.

Song Feng, Ritwik Banerjee, and Yejin Choi. 2012. Syntactic stylometry for deception detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 171–175. Association for Computational Linguistics.

Mark Frank and Paul Ekman. 1997. The ability to detect deceit generalizes across different types of high-stake lies. *Journal of personality and social psychology*, 72(6):1429.

Zoltán Gyöngyi, Hector Garcia-Molina, and Jan Pedersen. 2004. Combating web spam with trustrank. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 576–587. VLDB Endowment.

Trevor J Hastie and Robert J Tibshirani. 1990. *Generalized additive models*, volume 43. CRC Press.

Ipsos. 2012. Socialogue: Five Stars? Thumbs Up? A+ or Just Average? http://www.ipsos-na.com/news-polls/pressrelease.aspx?id=5929.

Nitin Jindal and Bing Liu. 2008. Opinion spam and analysis. In *Proceedings of the international conference on Web search and web data mining*, pages 219–230. ACM.

Nitin Jindal, Bing Liu, and Ee-Peng Lim. 2010. Finding unusual review patterns using unexpected rules. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1549–1552. ACM.

Thorsten Joachims. 1999. Making large scale svm learning practical.

Marcia K Johnson and Carol L Raye. 1981. Reality monitoring. *Psychological review*, 88(1):67.

Mark Knapp and Mark Comaden. 1979. Telling it like it isn't: A review of theory and research on deceptive communications. *Human Communication Research*, 5(3):270–285.

Jiwei Li, Claire Cardie, and Sujian Li. 2013a. Topicspam: a topic-model-based approach for spam detection. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguis-tics*.

Jiwei Li, Myle Ott, and Claire Cardie. 2013b. Identifying manipulated offerings on review portals. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, Wash*, pages 18–21.

Ee-Peng Lim, Viet-An Nguyen, Nitin Jindal, Bing Liu, and Hady Wirawan Lauw. 2010. Detecting product review spammers using rating behaviors. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 939–948. ACM.

Juan Martinez-Romo and Lourdes Araujo. 2009. Web spam identification through language model analysis. In *Proceedings of the 5th International Workshop on Adversarial Information Retrieval on the Web*, pages 21–28. ACM.

David Meyer. 2009. Fake reviews prompt belkin apology.

Claire Miller. 2009. Company settles case of reviews it faked. *New York Times*.

Arjun Mukherjee, Bing Liu, Junhui Wang, Natalie Glance, and Nitin Jindal. 2011. Detecting group review spam. In *Proceedings of the 20th international conference companion on World wide web*, pages 93–94. ACM.

Arjun Mukherjee, Bing Liu, and Natalie Glance. 2012. Spotting fake reviewer groups in consumer reviews. In *Proceedings of the 21st international conference on World Wide Web*, pages 191–200. ACM.

Arjun Mukherjee, Abhinav Kumar, Bing Liu, Junhui Wang, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013a. Spotting opinion spammers using behavioral footprints. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 632–640. ACM.

Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Natalie Glance. 2013b. What yelp fake review filter might be doing. In *Seventh International AAAI Conference on Weblogs and Social Media*.

Matthew L Newman, James W Pennebaker, Diane S Berry, and Jane M Richards. 2003. Lying words: Predicting deception from linguistic styles. *Personality and social psychology bulletin*, 29(5):665–675.

Alexandros Ntoulas, Marc Najork, Mark Manasse, and Dennis Fetterly. 2006. Detecting spam web pages through content analysis. In *Proceedings of the 15th international conference on World Wide Web*, pages 83–92. ACM.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T. Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 309–319.

Myle Ott, Claire Cardie, and Jeff Hancock. 2012. Estimating the prevalence of deception in online review communities. In *Proceedings of the 21st international conference on World Wide Web*, pages 201–210. ACM.

Myle Ott, Claire Cardie, and Jeffrey T. Hancock. 2013. Negative deceptive opinion spam. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Short Papers*, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

Myle Ott. 2013. Computational lingustic models of deceptive opinion spam. *PHD, thesis*.

Sinno Pan and Qiang Yang. 2010. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359.

Tieyun Qian and Bing Liu. 2013. Identifying multiple userids of the same author. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, Wash*, pages 18–21.

Paul Rayson, Andrew Wilson, and Geoffrey Leech. 2001. Grammatical word class variation within the british national corpus sampler. *Language and Computers*, 36(1):295–306.

David Streitfeld. 2012. For 2 a star, an online retailer gets 5-star product reviews. *New York Times.*, 26.

Alexandra Topping. 2010. Historian orlando figes agrees to pay damages for fake reviews. *The Guardian.*, 16.

Guan Wang, Sihong Xie, Bing Liu, and Philip Yu. 2011. Review graph based online store review spammer detection. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 1242–1247. IEEE.

Guan Wang, Sihong Xie, Bing Liu, and Philip Yu. 2012. Identify online store review spammers via social review graph. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(4):61.

Guangyu Wu, Derek Greene, Barry Smyth, and Pádraig Cunningham. 2010. Distortion as a validation criterion in the identification of suspicious reviews. In *Proceedings of the First Workshop on Social Media Analytics*, pages 10–13. ACM.

Kyung-Hyan Yoo and Ulrike Gretzel. 2009. Comparison of deceptive and truthful travel reviews. In *Information and communication technologies in tourism 2009*, pages 37–47. Springer.

Lina Zhou, Judee K Burgoon, Douglas P Twitchell, Tiantian Qin, and Jay F Nunamaker Jr. 2004. A comparison of classification methods for predicting deception in computer-mediated communication. *Journal of Management Information Systems*, 20(4):139–166.

Author Index